MASTER OF SCIENCE BY RESEARCH

Development and implementation of a new hybrid RANS/LES model for transitional boundary layers in OpenFOAM

Beechook, Abhi

Award date:
2015

Awarding institution:
Coventry University

Development and implementation of a new hybrid RANS/LES model for transitional boundary layers in OpenFOAM

Abhinivesh Beechook
Author

December 2015

School of Mechanical, Aerospace & Automotive Engineering
Faculty of Engineering, Environment & Computing
Coventry University
Priory Street
Coventry CV1 5FB

Development and implementation of a new hybrid RANS/LES model for transitional boundary layers in OpenFOAM

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science by Research (MScR) in Aerospace Engineering

Abhinivesh Beechook
Author

Dr. Humberto Medina
Supervisor (Director of Studies)

Mr. Remus Cirstea
Supervisor

December 2015

Coventry University

School of Mechanical, Aerospace & Automotive Engineering
Faculty of Engineering, Environment & Computing
Coventry University
Priory Street
Coventry CV1 5FB

Development and implementation of a new hybrid RANS/LES model for transitional boundary layers in OpenFOAM

Abhinivesh Beechook

Thesis for the degree of Master of Science by Research (MScR)

School of Mechanical, Aerospace & Automotive Engineering
Faculty of Engineering, Environment & Computing
Coventry University
Priory Street
Coventry CV1 5FB
United Kingdom

This document was typeset using LaTeX

Coventry University
United Kingdom, 2015

Abstract

Boundary layer transition occurs in a wide range of engineering applications and accurately modelling transition has been a challenge for over a century. In recent years, hybrid RANS/LES modelling approaches have gained significant attention by the research community. In essence, hybrid RANS/LES approaches employ Reynolds-averaged Navier-Stokes (RANS) in wall regions (i.e. to model the boundary layer) whilst applying subgrid-scale large-eddy simulation (LES) models to separated flow regions. A new hybrid RANS/LES model was developed as part of the current work for transitional boundary layers. The hybrid RANS/LES model was developed using the detached-eddy simulation (DES) hybridisation approach [1] and using the $k_T - k_L - \omega$ [2] transition model as the background RANS model. The model was implemented in OpenFOAM, an open source CFD package written in C++ programming language. The model development process starts with the formulation and implementation of the DES version of the hybrid RANS/LES model, the $k_T - k_L - \omega$ DES model. The model when tested on a 2D flat plate at zero-pressure-gradient, failed to capture laminar-turbulent transition, and the solutions remained fully laminar. This was due to a serious problem faced by the DES approach. In order to alleviate the problem faced by the DES model, the delayed detached-eddy simulation (DDES) approach was used to develop the $k_T - k_L - \omega$ DDES model, which responded very well when tested on the 2D flat plate configuration. The DDES results were compared with the RANS ($k_T - k_L - \omega$ transition model) results, the $k_T - k_L - \omega$ DDES demonstrated approximately a 30% improvement in the predicted transition location. The newly developed $k_T - k_L - \omega$ DDES model was also tested on a 2D cylinder and the results obtained showed reasonably good agreement with experimental data, which also indicated the need to calibrate the model for important parameters such as the DES model constant, C_{DES}.

Acknowledgements

I would like to express my gratitude to my Director of Studies, Dr. Humberto Medina, for the incredible amount of advice and support he provided during the course of this project. It has really been a rewarding experience to work with such an individual who is very passionate and has an "ocean" of knowledge about the subject.

I would also like to acknowledge Mr. Remus Cirstea, my 2nd supervisor, who has always been very helpful with any queries related to the project, together with his invaluable feedback and numerous useful discussions of CFD related matters.

Abhinivesh Beechook
December 2015

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

$2D$	two dimensional	
$3D$	three dimensional	
C_p	pressure coefficient	
C_μ	turbulent viscosity coefficient	
C_{DES}	detached-eddy simulation model constant	
C_f	skin friction coefficient	
C_S	Smagorinsky model constant	
D	diffusion term of RANS turbulence model	
D_L	anisotropic (near-wall) dissipation term for k_L	
D_T	anisotropic (near-wall) dissipation term for k_T	
f_ν	viscous near-wall damping function	
f_ω	boundary layer wake term damping function in ω equation	
$f_{\tau,l}$	time-scale damping function	
f_d	shield function	
f_{INT}	intermittency damping function	
$f_{NAT,crit}$	model function incorporating freestream turbulence effects on natural transition	
f_{SS}	shear-sheltering damping function	
f_W	inviscid near-wall damping function	
K	acceleration parameter of boundary layer transition	
k	turbulent kinetic energy	$(m/s)^2$
k_L	laminar kinetic energy	$(m/s)^2$
k_T	turbulent kinetic energy	$(m/s)^2$
$k_{T,l}$	effective large-scale turbulent kinetic energy	$(m/s)^2$
$k_{T,s}$	effective small-scale turbulent kinetic energy	$(m/s)^2$

k_{TOT}	total fluctuation kinetic energy, $k_T + k_L$	$(m/s)^2$
L	length	m
l	length scale	m
L/D	lift-to-drag ratio	
l_ν	viscous length scale	m
l_{DES}	DES length scale	m
l_{LES}	LES length scale	m
l_{RANS}	RANS length scale	m
P	production term of RANS turbulence model	
p	pressure	(N/m^2)
P_{k_L}	production of laminar kinetic energy by mean strain rate	
P_{k_T}	production of turbulent kinetic energy by mean strain rate	
Pr_θ	turbulent Prandtl number	
R_{BP}	by-pass transition production term	
r_d	boundary layer sensor function	
R_{NAT}	natural transition production term	
Re	Reynolds number	
Re_x	Reynolds number based on the streamwise distance	
Re_Ω	Reynolds number based on vorticity	
Re_τ	Reynolds number based on friction velocity	
Re_θ	Reynolds number based on momentum thickness	
Re_{crit}	critical Reynolds number	
Re_T	turbulence Reynolds number	
S	mean strain rate tensor magnitude	
S_{ij}	strain rate tensor	
T	temperature	K
t	time	s
T-S	Tollmien-Schlichting	
t_ν	viscous time scale	s
Tu	turbulence intensity	%
U	velocity	(m/s)
u	x-component of velocity	(m/s)
u_i	velocity vector	

u_τ	wall friction velocity	m/s
v	y-component of velocity	(m/s)
w	z-component of velocity	(m/s)
x	downstream distance	m
x_i	position vector	
y	distance to the nearest solid boundary (wall)	m
y^+	normalised wall distance in wall units	

Greek Symbols

α_θ	turbulent thermal diffusivity	
α_T	effective diffusivity for turbulence dependent variables	
β_{BP}	by-pass transition threshold function	
β_{NAT}	natural transition threshold function	
β_{TS}	Tollmien-Schlichting threshold function	
Δt	time step	s
Δ	filter width	m
δ	boundary layer thickness	m
ϵ	turbulence dissipation rate	(m^2/s^3)
η	Kolmogorov length scale	m
λ_{eff}	effective (wall-limited) turbulence length scale	m
λ_T	turbulent length scale	m
μ	dynamic viscosity	(kg/ms)
μ_T	turbulent or ‘eddy’ viscosity	(kg/ms)
ν	kinematic viscosity	(m^2/s)
ν_T	turbulent kinematic ‘eddy’ viscosity	(m^2/s)
$\nu_{T,l}$	large-scale turbulent viscosity contribution	(m^2/s)
$\nu_{T,s}$	small-scale turbulent viscosity contribution	(m^2/s)
ω	specific turbulence dissipation rate	s^{-1}
Φ_{BP}	model by-pass transition parameter	
Φ_{NAT}	model natural transition parameter	
ρ	density	(kg/m^3)
τ_{ij}	subgrid stress tensor	
τ_w	wall shear stress	(kg/ms^2)
θ	boundary layer momentum thickness	m

Superscripts

$+$	normalised quantity in friction wall units

Subscripts

∞	freestream value
ref	reference value of a quantity
rms	root mean square value of a quantity
sgs	subgrid-scale quantity
w	value at a solid boundary (wall)
x	x-component of a quantity
y	y-component of a quantity
z	z-component of a quantity

Symbols

$\overline{x}$	ensemble averaged x (RANS)
$\overline{x}$	filtered x (LES)
x'	fluctuation of x

Acronyms

CFD	computational fluid dynamics
CFL	Courant-Friedrichs-Lewy condition
CV	control volume
DDES	delayed detached-eddy simulation
DES	detached-eddy simulation
DNS	direct numerical simulation
ERCOFTAC	European research community on flow, turbulence and combustion
FST	freestream turbulence
FVM	finite volume method
GIS	grid-induced separation
HPC	high performance computer
IDDES	improved delayed detached-eddy simulation
LE	leading edge
LES	large-eddy simulation
LSB	laminar separation bubbles
LST	linear stability theory
MSD	modelled stress depletion

NACA	national advisory committee for aeronautics
NASA	national aeronautics and space administration
PISO	pressure-implicit with splitting of operators
RANS	Reynolds-averaged Navier-Stokes
SA	Spalart-Allmaras
SGS	subgrid-scale
SIMPLE	semi-implicit method for pressure linked equations
SIMPLEC	semi-implicit method for pressure linked equations consistent
SST	shear stress transport
UAV	unmanned aerial vehicle
URANS	unsteady Reynolds-averaged Navier-Stokes
ZPG	zero-pressure-gradient

Chapter 1

Introduction

Boundary layer transition has been rigorously studied for over a century and despite the significant progress made, it has been extremely challenging for scientists and engineers to derive a theory that is valid for all types of transition mechanisms. As a result of the absence of a universally valid theory for the transition phenomenon, attempts to develop sophisticated transition prediction models have been a significant and active area of research. This is primarily because the complex fluid flow phenomenon of boundary layer transition is present in a wide range of engineering applications, for instance, in the case of Unmanned Aerial Vehicles (UAVs) the effect that laminar separation bubbles (LSB) have on the transition process is not fully understood. Other examples include the limited understanding of how the Coriolis effect and the three-dimensional nature of the flow over wind turbines affects the transition process, or how receptivity and system vibration affects the boundary layer transition process on helicopter blades, and this is by no means an exhaustive list.

The first transition models developed were based on Linear Stability Theory, the e^N method and experimental correlations. These early transition prediction methods, especially the e^N method, are very useful. However, these models have disadvantages such as their non-universal calibration, making them highly dependent on experimental data. This highlights the strong argument for the development of alternative transition prediction approaches using modern simulation techniques, namely Computational Fluid Dynamics (CFD), which is currently the most widely used method in numerous sectors (aerospace, automotive, meteorology, etc.). Direct numerical simulation (DNS), large-eddy simulation (LES) and Reynolds-averaged Navier Stokes (RANS) are the three major CFD modelling approaches suitable for transitional/turbulent flows. The RANS method is currently the most widely used CFD method, and in fact, is considered as the industrial "backbone" to flow computations. Despite several years of profound effort, the accuracy of the RANS method is still questionable (particularly in relation to separated flows). However, RANS models for near-wall turbulence not involving flow separation have matured significantly and offer reasonable accuracy. Alternatively DNS is an approach that can offer significant information about the flow field, since it resolves all the turbulent scales unlike the modelling approach of RANS (i.e. all the characteristics of turbulence are modelled). However, DNS is principally limited to low Reynolds number (Re) applications, and is computationally expensive since it requires extremely fine grids for correctly resolving the relevant turbulent length scales. The LES approach is concerned with resolving the large scales and modelling the small scales of turbulence. LES is primarily used as a research tool rather than an industrial tool due to its prohibitive computational cost associated with high Reynolds number flows, especially for wall-bounded flows. As a

result of these deficiencies of the primary CFD methods, i.e. RANS, LES and DNS, there has been an outbreak of new CFD approaches, which are categorised as hybrid RANS/LES methods (a detailed review of which is provided in Chapter 2). Hybrid RANS/LES models involve both the RANS and LES turbulence modelling techniques, providing a compromise between the accuracy of LES and the cost effectiveness of RANS. Whilst numerous hybrid RANS/LES models have been developed for turbulent flow applications, there has been limited development of such models towards the aim to addressing the complex problem of transitional boundary layers and transition prediction.

Therefore, within the scope of this work, a hybrid RANS/LES turbulence model for transitional boundary layer flows has been formulated, implemented and numerically tested in OpenFOAM. OpenFOAM is an open source CFD toolbox, based on the finite volume method (FVM). It is essentially a collection of libraries with functionality, written in C++, which can be utilised to solve CFD problems. An introduction to OpenFOAM is presented in section 4.1. The hybrid model was developed using an approach following the principles of the DES methodology proposed by Spalart et al. [1], who pioneered the hybrid RANS/LES modelling approach. The three-equation ($k_T - k_L - \omega$) eddy viscosity transition model of Walters and Cokljat [2] was used as the underlying RANS model for the formulation of the new DES model implemented since it has been reported to perform well in predicting boundary layer transition (discussed in more detail in Chapter 2). The implemented hybrid RANS/LES model was numerically tested on 2D canonical cases (zero-pressure-gradient flat plate and cylinder) to verify the numerical implementation of the model.

1.1 Aim & objectives

1.1.1 Aim statement

To develop a suitable formulation for a new hybrid RANS/LES model and implement it in OpenFOAM. The model will be based on the $k_T - k_L - \omega$ RANS model of Walters and Cokljat [2] in order to incorporate the capability of this model to predict boundary layer transition.

1.1.2 Specific objectives

1. To critically review the current state-of-the-art of the classic (DNS, LES and RANS) and hybrid turbulence modelling techniques used in CFD when applied to the prediction of boundary layer transition in order to identify and justify a new hybrid RANS/LES model formulation (in terms of the predictive capabilities of existing models, their key features and limitations, and recent breakthroughs).

2. To assess and validate a selection of RANS turbulence models available in OpenFOAM for a 2D flat plate configuration at zero-pressure-gradient, with a particular focus on the $k_T - k_L - \omega$ transition model (benchmarked against the ERCOFTAC [3] flat plate experimental dataset).

3. To investigate the suitability of various inlet conditions, simulation strategies and a selection of LES models for the prediction of boundary layer transition on a 2D flat plate at zero-pressure-gradient.

4. To formulate and implement a new hybrid RANS/LES turbulence model in OpenFOAM, and to test the implementation on 2D canonical test cases (zpg plate and cylinder).

1.2 Thesis outline

This thesis consists of eight Chapters, headed by this introduction to the work.

Chapter 1 provides a succinct background to the importance of transition modelling and its relevance to engineering applications, followed by a brief explanation of the purpose(s) of this work. The aim and objectives of this work are also discussed within this Chapter.

Chapter 2 presents the literature review highlighting the importance of transition modelling and a brief description of the early transition modelling approaches. The various CFD approaches to transition prediction are described, providing an overview of the state-of-the-art. The advantages and disadvantages of each of the CFD approaches when applied to boundary layer transition prediction are highlighted, supporting the need to develop new CFD modelling approaches and the motivation of the current work on the development and implementation of a hybrid RANS/LES model with laminar-turbulent transition capabilities.

Chapter 3 provides an overview of the flow physics relevant to this work, with a description of the relevant governing equations, and the RANS turbulence models and SGS models that were tested as part of the methodology in the development of the proposed hybrid RANS/LES model.

The numerical aspects of turbulence modelling are described in Chapter 4, which includes key information about the flow solver and the finite volume discretisation methods, as well as a summary of the test cases carried out as part of this work.

Chapter 5 presents a detailed assessment of a range of selected high and low Reynolds number models. The $k_T - k_L - \omega$ transition model of Walters and Cokljat [2] was also tested on a two-dimensional canonical zero-pressure-gradient flat plate to study their performance in capturing the complex laminar-turbulent physics. The detailed numerical setup and various boundary conditions for each tested model are provided, along with a thorough discussion of the performance of the different models, through direct comparison with the experimental data of zero-pressure-gradient ERCOFTAC [3] T3 series.

As the work involves developing a hybrid RANS/LES model which involves subgrid-scale model characteristics, it was very important to have a prior knowledge of the SGS models behaviour, particularly in the context of transition prediction. Chapter 6 covers the LES computational investigations relevant to this work. Taking into account one of the most challenging practices of LES, i.e. selecting an appropriate inflow boundary condition, an assessment of various LES inflow conditions was performed to evaluate their performance. A LES grid resolution study was also carried out to identify grid requirements to capture transitional flow characteristics on a 2D/3D zero-pressure-gradient flat plate.

Chapter 7 provides a background of the adopted hybrid RANS/LES method for the development of the proposed model. The detailed mathematical formulation and implementation of the new hybrid RANS/LES model are also presented, together with the numerical setup and boundary condition specifications. Ultimately, the results from these numerical tests, performed on a 2D zero-pressure-gradient flat plate and a 2D cylinder are presented, which were used to verify that the model was implemented correctly and is fully operational.

Chapter 8 presents a conclusion summarising the outcomes of the work, the key contributions and a list of potential future work.

Chapter 2

Literature review & motivation

Boundary layer transition is a complex process which must be considered when designing systems within a wide range of engineering applications, including aerospace, automotive, power generation, heating and cooling, biomedical, marine systems and chemical processing. Essentially, boundary layer transition is the process by which a laminar boundary layer becomes turbulent (a review of boundary layer transition is included in section 3.2). Transition has a direct effect upon important flow parameters such as wall shear stress distribution and surface heat transfer. Therefore, in many cases such as transition to turbulence in aircraft, where turbulent boundary layers are linked is linked to higher skin friction, it is of paramount importance to be able to accurately predict the onset of transition as well as capturing the non-linear interactions that lead to a fully turbulent in order to obtain accurate skin friction predictions (and subsequent fuel consumption estimates).

Due to its practical importance, various approaches have been developed in order to model and predict transition. Early transition models were developed based on the e^N method and Linear Stability Theory [4], together with experimental correlations [5, 6]. Although these transition modelling/prediction approaches have been very useful and successfully applied to a range of engineering configurations, they strongly rely on experimental data for calibration and are not universal [7]. When these models are used to evaluate complex systems, such as those involving flow control, correlation-based models can be expensive due to the need to develop suitable experimental programmes to support their re-calibration. Therefore, numerical methods and models that do not involve or rely on experimental correlations are generally favoured. In CFD, Direct numerical simulation (DNS), large-eddy simulation (LES) and Reynolds-averaged Navier-Stokes (RANS) approaches are considered feasible alternatives to correlation-based methods. Each of these three techniques have advantages and disadvantages. In order to identify their applicability to transition modelling, these three simulation techniques will be critically reviewed.

DNS is considered as the most "straightforward" CFD approach for the numerical study of turbulent flows [8]. DNS provides the solution of the flow field using the Navier-Stokes equations directly. Therefore, there is no requirement for employing a turbulence model to close the system of equations. DNS is, no doubt, the method available for the computational study of turbulence/turbulent which provides th most information about flow structures[9]. The very first DNS computations dates back from the early 1980s, during the time when computing power started to significantly improve [10]. One of the first DNS contributions include the pioneering work carried out by Fasel [11], which focused on two-dimensional flows involving small amplitude instabilities.

Fasel and Bestek [12] successfully applied the DNS approach to the study of non-linear, spatial disturbance amplification in plane Poiseuille flow. The first well-resolved flow simulation using DNS was performed by Gilbert [13] and Gilbert and Kleiser [14]. Their work involved the computation of three-dimensional transition to turbulence where Klebanoff type (K-type) transition was simulated in plane channel flow. Since then, the DNS approach continues to be successfully applied to boundary layer transition [9, 15–17]. Rai and Moin [18] simulated the complete transition of a spatially growing boundary layer using DNS along with various grid densities within different regions of the computational domain. They used a coarse grid in the transitional region, a fine grid in the turbulent region and, again, a coarse grid near the outlet in order to reduce computational power requirements. Their results showed good agreement with experimental results which proves the feasibility of simulating transition using DNS. Sayadi et al. [19] employed DNS to compute Klebanoff type (K-type) and Herbert type (H-type) transitions on a spatially evolving zero-pressure-gradient flat plate boundary layer. The amplitudes of velocity fluctuations of the numerically computed H-type transition when compared with the experimental results of Kachanov and Levchenko [20] showed good agreement. Additionally, the velocity fluctuation amplitudes within the transitional boundary layer of the K-type transition when compared with the computational results of Rist and Fasel [21] showed very good agreement.

DNS is regarded as a suitable tool for transition prediction even though the specification of suitable inlet conditions for the external disturbance level and structural flow features remains one of the greatest computational challenges. In principle, the stages of transition, i.e. the breakdown of laminar flow, the formation of turbulent spots and the transition to fully turbulent flow, can be simulated with high precision using the DNS approach [22]). The major disadvantages of DNS are that it requires an excessively large amount of computing power and high order numerical schemes to ensure accurate solutions. Additionally, the computational domain has to be reasonably large and the grid has to be sufficiently dense in order to ensure that all the dissipation length scales, which are on the order of the Kolmogorov length scale (η), are captured. The domain size depends on the integral length scale and the number of grid points is a function of the flow Reynolds number, *Re*. As *Re* increases, the integral to smallest length scale ratio in the flow increases. The computational grid must have the capability to capture these smallest scales of motion [23]. Zheng et al. [24] performed DNS simulation of a flat plate transitional boundary layer that took approximately four weeks on a supercomputer with 64 processors to compute the solution with approximately 5×10^7 grid points. Wu and Moin [25] simulated by-pass transition induced by freestream turbulence using DNS with approximately 2×10^8 grid points. The study of boundary layer transition by Rai and Moin [18] required 800 hours to compute the solutions. The DNS of transition performed by Sayadi et al. [19] required approximately 1 billion grid points for each simulation. All of these are examples of the high computational cost of the DNS approach. Despite the swift increase in computing power, the high computational cost of DNS is still prohibitive, which as a result limits its use mainly to research purposes. As a result of its computational resources requirement, DNS was deemed inappropriate for the current work.

At the other extreme of the methodological scale to DNS, turbulence can be statistically described by mathematical models which rely on the use of the Reynolds-averaged Navier-Stokes (RANS) equations. In the RANS approach, the turbulent stresses are modelled based on the classical assumption that turbulent motion is almost in a state of equilibrium, assuming isotropic turbulence and based on the Boussinesq hypothesis. The RANS approach operates by time-averaging of the main flow field so as to combine all the fluctuating components into the "Reynolds stresses", which are required to be modelled. This approach to fluid flow computations requires a turbulence model to close the system of equations and for the approximation of the Reynolds stresses using the turbulent eddy viscosity [23]. The RANS method is applicable to transitional flow mod-

elling with significantly lower computing power requirements and lower computational cost than DNS and LES. Low Reynolds number RANS turbulence models are relatively simple approach to modelling laminar-turbulent transition and they have previously been successfully applied [26–31]. However, a recent investigation carried out on the performance of a range of low Reynolds RANS models suggested that their ability to model transition is purely a numerical artefact rather than a characteristic of any actual transition predictive capability [32]. Nevertheless, the RANS approach is still considered as the industrial workhorse for turbulence modelling as it represents a fair compromise between computational cost and accuracy.

Taking advantage of its cost effectiveness characteristic and its applicability to industrial configurations, several authors have devoted enormous time and effort towards the development of newly improved RANS transition models over the past years[2,33–35]. The common RANS approaches that researchers have been exploring include correlation-based modelling and transport equation modelling approaches. Correlation-based approaches involve coupling fully turbulent models with empirical transition correlations from experimental data. These correlation-based models [36,37] have been considered as useful tools in industry for transition modelling. However, correlation-based models are based on, and require, the specification of integral boundary layer parameters (such as the downstream distance from the leading edge of an aerofoil or the local momentum thickness of the boundary layer) which makes their implementation in most CFD packages challenging, particularly for complex 3D geometries. The transport equation modelling approach involves employing additional transport equations to existing turbulence model to incorporate the effects of transition. The models developed based on this approach include the intermittency model of Suzen and Huang [37] and the correlation based model of Menter et al. [38]. However, these models still require the specification of the integral parameters of the boundary layer in the simulation, thus restricting their applicability/use to limited case geometries/configurations. Lately, the focus of researchers has moved towards enhanced single-point RANS modelling techniques with the aim of developing transition models that will eliminate the requirement for integral quantities (such as momentum thickness) making them more "implementation friendly". The recent models developed include the $k-\omega-\gamma$ model of Fu and Wang [34] and the $k-k_L-\omega$ model of Walters and Leylek [35,39], which have been successfully implemented in commercial and open source CFD codes and are currently being tested to determine their transition prediction capability as reliable RANS transition models. In 2008, an improved version of the Walters and Leylek transition model was proposed by Walters and Cokljat [2], the $k_T-k_L-\omega$ model, which now includes the effect of shear-sheltering. In 2014, Medina and Early [7] proposed a modified version of the original $k_T-k_L-\omega$ model, rectifying errors in the original description of the model [2] which lead to an underestimation of friction in the turbulent region. They also introduced a novel function to the transition model so as to simulate the effects of aft-facing steps on the transition to turbulence. The $k_T-k_L-\omega$ model have been tested on various geometries and results showed very good agreement with the experimental data [40–44]. The in-house implementation of the $k_T-k_L-\omega$ used in [7] has been used as part of this current work. A detailed description of the $k_T-k_L-\omega$ model is provided in section 3.3.4.4.

Despite the extensive use of RANS models, the main limitation is that they provide a statistical description of the flow, requiring the averaging of the flow field. RANS models also fail to differentiate between quasi-periodic large scale and small scale features of the flow field. This is considered to be problematic when the flow field is governed by both phenomena. A typical example is the flow past a cylinder. The RANS modelling approach fails to accurately replicate the unsteady features of the flow field, for instance, vortex formation and shedding. Despite using the most advanced turbulence models, that are generally semi-empirical consisting of numerous empirical constants, it is still a challenge for the RANS technique to accurately predict these large

scale flow characteristics [45]. In terms of transition prediction, with the exception of the work of Walters and Cokljat [2], most RANS models do not have the predictive capability of the transition point location in the boundary layer since they require *a priori* knowledge of the "location" of the laminar-turbulent transition [19]. Furthermore, Peneau et al. [46] argued that RANS CFD simulations fail to provide a strong confirmation of the separation point when computing aerodynamic properties of aerofoils, especially at stall angles. This is due to the fact that RANS based turbulence models have limited flow separation modelling capabilities. Despite the recent development of advanced RANS transition models, for instance, the $k_T - k_L - \omega$ model, the problem of transition modelling is still not fully solved, especially when the model is applied to configurations involving flow separation. Therefore, alternative CFD approaches for accurately predicting laminar-turbulent transition for separated flows have to be identified.

Large-eddy simulation, also referred to as the "partial resolution of turbulence method", is a viable alternative approach to fluid flow computation. The LES technique, first proposed by Joseph Smagorinsky [47] in 1963, was developed as an attempt to simultaneously improve the accuracy and applicability of RANS and minimise the restrictive cost of DNS. In LES, the Navier-Stokes equations are the governing equations but, unlike the RANS approach, spatial filtering is applied instead of time-averaging and the turbulent stresses are divided into resolved and modelled stresses. The dynamics of the large scales of turbulence are resolved, overcoming the accuracy issue of the RANS approach in regions of flow separation, while the computationally intensive small scales are modelled using a subgrid-scale (SGS) model. This operational mechanism of the LES approach makes it more computationally affordable when compared to DNS. The practice of LES dates back to the 1970s and was first explored by Deardorff [48]. Since then, and taking into account the increase in computing power, LES is now considered as a feasible alternative to fluid flow computations.

Although the application of LES to turbulent flows has proved to be effective over the past decades, its application to transitional flows has been fairly recent. LES computations of laminar-turbulent transition in temporally developing boundary layer and channel flow were carried out by Piomelli et al. [49] and Piomelli and Zhang [50] and the numerical results revealed that the eddy viscosity at the initial stages of transition has to be extremely low (almost inactive) in order to allow the proper development of flow perturbations. Yang et. al [51] studied by-pass transition of flow over a flat plate using LES, with FST level of 5% and a modified version of the Smagorinsky SGS model in order to avoid excessive dissipation in the laminar boundary layer. Their numerical results showed good agreement with experimental data of skin friction coefficient but the simulations failed to demonstrate the formation of turbulent spots. This is a known limitation of the Smagorinsky model, which is over-dissipative causing the flow to remain laminar. This problem is overcome by the dynamic procedure proposed by Germano et al. [52], which is described in section 3.3.5.4. Voke and Yang [53] carried out LES of by-pass transition by applying a low Reynolds number correction to the original Smagorinsky SGS model and the results indicated good agreement of the numerical flow characteristics with the experimental flow characteristics. Huai et al. [54] simulated laminar-turbulent transition over a spatially developing flat plate boundary layer, employing the dynamic procedure of Germano et al. [52] to avoid the problem posed by the standard Smagorinsky SGS model and the results demonstrated very good agreement with the experimental data, accurately predicting the appropriate growth of perturbations in the early stages of transition, together with the shear layers and vortical structures in the laminar breakdown stage and the turbulent statistics in the turbulent regime. Over the recent years, there has been numerous other successful applications of LES to transitional flows [10, 55, 56]. A classic LES study relevant to the current study is the application of LES to natural transition [56]. LES of natural H-type and K-type transitions was carried out, testing a range of SGS models on dif-

ferent grid resolutions. The results indicated that the standard constant coefficient Smagorinsky model fails to predict the transition location as demonstrated by the deviation of the skin friction coefficient when compared with the experimental data. The study [56] also provided a firm that dynamic SGS LES models can be emplyed to predict boundary layer transition.

Although DNS has the capability to resolve the complete set of turbulent scales, its high computational cost makes this method infeasible at present. The RANS method less computationally expensive in comparison to DNS, but its universality is uncertain as a result of modelling approximations. LES is positioned in between the extremes of DNS and RANS. LES is still not routinely applied by industry as a result of (still) high computational cost due to the near-wall region (attached boundary layers), where the turbulent scales are very small and comparable to the Kolmogorov scales, requiring extremely fine computational grids, approaching that of DNS. Spalart et al. [1] performed a computational cost analysis of the LES for an airplane wing and it was found that for a flow of Reynolds number of order 10^7, the computational grid has to consist of at least 10^{11} cells and the number of time steps has to be of order 5×10^6. Boundary layer growth and separation and the transfer of momentum within the region of the separated flow have been classed as the two major computational challenges when simulating turbulent flows [57]. In order to overcome these computational difficulties hybrid RANS/LES modelling approaches have been introduced that essentially involve a combination of RANS and LES attributes using a single turbulence model. Typically, hybrid RANS/LES modelling approaches provide significantly more accurate solutions than RANS and at a lower cost than LES. The RANS approach capable of modelling boundary layer growth and the LES approach is applicable to the momentum transfer within separated regions. Nevertheless, neither RANS nor LES have been evidenced to independently solve both of these problems at reasonable cost/accuracy.

Hybrid RANS/LES modelling approaches can be classified as zonal and non-zonal. In the zonal hybrid RANS/LES modelling approach, a RANS turbulence model is applied in the user-defined region(s) within the computational domain, while applying a SGS model to the rest of the computational domain. The zonal hybrid RANS/LES approach involves solving two different eddy viscosity equations (one for RANS and one for LES), followed by combining the RANS and LES fields at a pre-defined region within the domain by matching a secondary variable, for instance, the eddy viscosity or the wall shear stress. Zonal hybrid RANS/LES models experience the problem of switching from the RANS to LES region, which as a result is still a research intensive subject [58, 59]. In the non-zonal approach, a single eddy viscosity equation is used for RANS and LES, for which the switch is initiated by a simple modification of the length scale of the destruction term within the eddy viscosity equation. As a result of its simplicity, non-zonal hybrid RANS/LES models are "implementation friendly", but they prevent the user from pre-defining RANS or LES regions within the computational domain (the only means the user has to select the operation mode e.g. RANS or LES, is through the density of the mesh used in the computations). The detached-eddy simulation (DES) model of Spalart et al. [1] is one of the most widely used non-zonal models. In the DES model, the switch from the RANS to the LES mode and vice-versa is based upon the local grid size. The original DES model is based on the original Spalart-Allmaras model [60] which changes to an one-equation SGS model when operating in LES mode. Subsequent to the pioneering work of Spalart et al. [1], the hybrid RANS/LES approach has witnessed an unceasing interest, with a wide range of alternative hybrid RANS/LES models developed [61–64]. The hybrid RANS/LES approach of Spalart et al. [1] has been extensively adopted to develop new DES models using other RANS models. For example, Jee and Shariff [65] from NASA Ames Research Center recently proposed a new DES model based on the $v^2 - f$ model. Unlike other RANS models used in DES, the $v^2 - f$ model incorporates anisotropy of near-wall turbulence. The $v^2 - f$ model was modified so that the DES model re-

duces to a transport equation for the subgrid-scale kinetic energy in isotropic turbulence. The $v^2 - f$ based DES model was tested around a cylinder at Re = 3900 where the flow separates. The computational results indicated that this model represents the turbulent wake as accurately as the dynamic Smagorinsky model [65]. Strelets [66] presented a DES model based on Menter's $k - \omega$ SST model [67], which was developed mainly for the application to massively separated flows. The results obtained by Strelets were very promising. Other DES models have been developed, details of which can be found in [68]. These existing variants of DES models demonstrate the feasibility of developing DES turbulence models based on other RANS models.

The majority of the developed hybrid RANS/LES models are based on standard turbulent eddy-viscosity RANS models and they cannot predict laminar-turbulent transition. Therefore, employing a transition sensitive RANS model in the development of hybrid RANS/LES models for transitional flows applications may be promising [69]. In fact, to the best of the author's knowledge, to date there exist only two hybrid RANS/LES models that were developed specifically for transitional applications. Sørensen et al. [69] formulated hybrid RANS/LES model using the DES approach, based on the correlation-based transition model, $\gamma - Re_\theta$ of Menter et al. [70, 71]. Their model was tested on a cylinder case and a thick aerofoil section (DU-96-W-351). Their results proved that the adopted DES methodology for the development of the hybrid RANS/LES model to be computationally robust, demonstrating the capability of predicting laminar separation, turbulent re-attachment and turbulent separation. However, this model inherits the weaknesses associated with correlation-based RANS models that were previously discussed. Additionally, Alam et al. [72] proposed a transition-sensitive hybrid RANS/LES modelling methodology using the monotonically-integrated large-eddy simulation (MILES) approach [73], which involves the formulation and development of a dynamic hybrid RANS/LES model based on the physics of the $k_T - k_L - \omega$ transition model of Walters and Cokljat [2]. The model was implemented in the commercial CFD package, ANSYS Fluent, and has been tested on a PAK-B aerofoil at a range of Reynolds numbers and freestream turbulence intensity (Tu) levels ($Re = 2.5 \times 10^4$ for low Tu levels and $Re = 1.0 \times 10^5$ for high Tu levels). The delayed detached-eddy simulation (DDES) model of Spalart et. al [74] was also tested for the same case for comparison purposes. The computational results from their proposed dynamic hybrid RANS/LES model when compared with experimental and DDES results reported the potential of transition-sensitive hybrid RANS/LES model for capturing the physics of the complex laminar-turbulent transition phenomenon [72].

In this current work, a hybrid RANS/LES model for transitional boundary layers is proposed. The motivation behind the development of such a model is due to the paramount importance of accurate laminar-turbulent transition prediction which is experienced in a wide range of engineering applications. A hybrid RANS/LES modelling framework was chosen as a result of the accuracy limitation of a pure RANS approach and the high computational cost associated with a pure LES approach. The development of the proposed hybrid RANS/LES model is primarily aimed towards the open source and research communities, which are considered as the immediate users. Besides, the model was developed on the basis that it can be used as a potential industrial tool in the future. The proposed hybrid model will be developed based on the DES approach [1] since a range of DES based hybrid RANS/LES models has been successfully developed[2] and tested, proving that this particular approach is computationally robust and simple to implement, in comparison to other existing hybrid RANS/LES modelling approaches. The hybrid RANS/LES model that will be developed for this current work is different from the hybrid models of Alam et al. [72]

[1] A detailed description of the DES hybrid RANS/LES modelling approach, together with its implementation is described in Chapter 7

[2] The interested reader is referred to the DESider reference [68], where a range of developed DES models based on different RANS model are described in detail

and Sørensen et al. [69]. The transition-sensitive hybrid RANS/LES model developed by Alam et al. [72] employed the $k_T - k_L - \omega$ transition model as the RANS model and a dynamic subgrid-scale model. The presence of the dynamic part within the model increases the computational time at every time step since an additional calculation is required to determined the model coefficients. The transition-sensitive hybrid RANS/LES model developed by Sørensen et al. [69] employed the $\gamma - Re_\theta$ correlation-based transition model as the background RANS model for the development of a transition sensitive DES model.

Although the employed hybridisation approach for the proposed hybrid RANS/LES model is the detached-eddy simulation (DES), which is the same approach used by Sørensen et al. [69], the proposed model employs the $k_T - k_L - \omega$ transition model [2] which does not require of experimental correlations. Furthermore, using the DES approach does not require the dynamic procedure to be implemented, which makes the model more computationally economic in comparison to the dynamic hybrid RANS/LES model of Alam et al. [72].

It is to be noted that the scope of this work is limited to the hybrid RANS/LES model formulation, implementation and numerical tests on simple canonical configuration of a 2D flat plate at zero-pressure-gradient and a 2D cylinder test case. The numerical tests performed on the 2D flat plate and cylinder test cases are intended solely to verify the model implementation. The model will need to be thoroughly validated and calibrated using experimental data for a range of geometries and configurations. Finally, boundary layer separation, although cited and described throughout this thesis, is not within the scope of this work.

Chapter 3

Flow physics & turbulence modelling

The prediction of turbulence poses a challenge to the modelling capabilities of Computational Fluid Dynamics (CFD). From the time since research in computational methods of turbulence began, there has been a series of methodologies aiming to integrate the effect of turbulence in computational simulations of fluid flow, each with their corresponding advantages and disadvantages. These methods will be discussed in this Chapter, along with the relevant flow physics of turbulent and transitional flows that underpin this work.

Turbulence is defined as a three-dimensional time dependent motion involving vortex stretching, which as a result, causes velocity fluctuations to spread to all wavelengths [75]. The behaviour and physical properties of a fluid is very important to study for many engineering applications. Very often, it is of fundamental importance to identify whether the flow is laminar, transitional or turbulent in nature. Laminar flow is referred to as a well ordered and structured fluid flow which typically flows in parallel layers without any disturbance between the layers, i.e. there are no eddies perpendicular to the flow direction. A fluid undergoing turbulent flow exhibits fluctuations (of all flow variables) and is characterised by stochastic property changes. Turbulent flows usually occur once a characteristic critical Reynolds number (Re_{crit}) is exceeded, leading to the amplification of flow instabilities and the subsequent development of turbulent flow The process when a laminar flow develops to turbulent flow is termed "laminar-to-turbulent transition".

As stated in Chapters 1 and 2, this work is focused primarily on the development of an improved numerical modelling approach (Hybrid RANS/LES) for laminar-turbulent transition and therefore, it is prudent to introduce the concept of boundary layer transition. Laminar-turbulent transition is considered to be one of the most complex phenomenon in Fluid Mechanics. Modelling transition using one of the most advanced numerical methods, CFD, still remains a challenge despite the ongoing development of new turbulence and transition models, and the steady increase in computing power. Several experimental and numerical methods have been developed in an attempt to accurately predict transition. A general overview of boundary layer theory will be presented, followed by the laminar-turbulent transition process. The different modes of boundary layer transition will also highlighted together with the main factors affecting the process of laminar-turbulent transition. Additionally, the various CFD modelling approaches will be outlined, together with a quick overview on the relevant turbulence models and their modelling requirements.

3.1 Overview of the boundary layer concept

Ludwig Prandtl, a German engineer, proposed the boundary layer concept in 1904 [76] which had a profound effect on our understanding of fluid mechanics. Prandtl's boundary layer concept revolutionised how scientists conceptualised the motion of fluids. Before Prandtl, the function of viscosity in a fluid was unclear and the viscous effects were neglected in fluid flow. The exclusion of viscous effects resulted in a simplified derivation stating that in a steady flow the aerodynamic drag on a body is zero. In 1904, Prandtl proved that the viscous effects should not be neglected in fluid flow computations, regardless of how small the magnitude of the fluid's viscosity[1] is. Prandtl noted that when modelling/predicting the flow field, the viscous interactions away from the solid boundary are not as critically important as those [the viscous interactions] close to the solid boundary. Therefore, at the wall, viscosity has to be taken into account since it has a major influence on the fluid flow. In 1908, four years after the introduction of the boundary layer concept and the laminar flow equations, Blasius, a German fluid dynamics physicist, provided the solutions of the Prandtl's equations for the skin-friction over a flat plate with a zero-pressure-gradient along its length [77]. The acceptance of Prandtl's proposed concept was delayed for about 20 years [78] until the evolution of experimental techniques to the point where it was possible to explore the inner structure of the boundary layer at a detailed level.

When an object is in motion travelling through a fluid (or vice versa), aerodynamic forces are generated. The magnitude of the resultant aerodynamic forces depends upon the shape of the object, the velocity and the fluid mass travelling by the object. The fluid particles directly next to the object sticks to the surface and the velocity approaches zero. Due to the effect of this complex movement of the fluid particles close to the object, a thin layer dominated by viscous effects is formed, very close to the surface of the object. Progressing away from the object's surface, the velocity increases from zero (initial velocity at the surface) to the freestream velocity, U_∞. The perpendicular distance from the surface to the point where the velocity reaches 99% of $U\infty$ bounds the region where viscous effects are most marked, and it is known as the boundary layer.

A fundamental parameter when dealing with boundary layers is the Reynolds number. It was discovered and established by Osborne Reynolds in 1883 [79], which can be used to characterise the point at which the flow of a typical fluid transitions from the laminar to turbulent regime. The Reynolds number is a dimensionless number, which can also be interpreted as the ratio of inertial to viscous forces for a given flow-object configuration as it requires a representative (or reference) length scale. The Reynolds number can be mathematically expressed as

$$Re = \frac{U_{ref} L_{ref}}{\nu} \tag{3.1}$$

where U_{ref} is the reference velocity and L_{ref} is the reference length and $\nu = \mu/\rho$ is the kinematic viscosity. At low Reynolds numbers (below the critical Reynolds number) small agitations (if any present) are damped out by the viscous stresses which prevent the flow from becoming turbulent. At high Reynolds number (above the critical Reynolds number), the viscous stresses are lower compared blueto the inertial agitations in the flow, which amplify and therefore, the flow is classified as turbulent.

[1] Viscosity is a measure of a fluid's resistance to motion. It characterises the internal frictional force of a fluid in motion.

3.2 Boundary layer transition

It has been over a century since the classical experiments performed by Osborne Reynolds with a water tank demonstrated the laminar-turbulent transition phenomenon in fluid flow. Since then, the subject of laminar-turbulent transition has maintained an unceasing research interest. The stages through which transition occurs may vary in different applications [33]. The development of a laminar flow along a given object is strongly influenced by different types of perturbations related to the freestream condition or to the object itself. Taking into account the sources of the perturbations, there are different paths that may lead to transition. When a laminar flow develops and evolves along a "hydraulically smooth" surface, for example, the skin surface of a Boeing 737 [80], where the freestream perturbation is very low, transition involves a range of well defined stages, i.e. starting from the linear amplification of fluctuating waves. This mechanism is known as natural transition. Another key path leading to turbulence is known as by-pass transition. By-pass transition occurs when a laminar boundary layer develops in the presence of strong perturbations (high freestream turbulence) which results in the omission of the initial stages of natural transition and eventually leading to turbulence. Other important types of laminar-turbulent transition that can be experienced in various applications, identified by Mayle [81], are namely separated flow transition and reverse transition. Although there are other transition types, the primary focus of this research will be on natural and by-pass transition. Despite the relatively few engineering applications where natural transition is a major concern, the stages involved are relevant to other types of transition, for instance, natural transition is important in aircraft flight at high altitudes where freestream turbulence levels are low. By-pass transition is experienced in a wide range of engineering applications, for example, in turbomachinery and gas turbine engines.

Figure 3.1 graphically represents the definition of "hydraulically smooth". For a hydraulically smooth surface, a reduction in roughness does not translate into a further reduction in skin friction. For a typical aircraft, its surfaces can be considered as "hydraulically smooth" [80].

Figure 3.1: Graphical representation of a hydraulically smooth surface [80]

With reference to figure 3.1, the illustrated results from studies carried out by Boeing concluded that on a typical Boeing commercial aircraft, the finished skin surfaces are basically hydraulically smooth. The equivalent sand grain roughness of the skin surface of a Boeing aircraft is well below 400 microinches ($1.016 \times 10^{-2} mm$).

3.2.1 The transition process

The transition process comprises of a series of stages, that can be explained by considering the canonical case of Blasius flow over a flat plate. Figure 3.2 illustrates the various stages during which a laminar flow transitions to turbulence, on a flat plate at zero-pressure-gradient.

Figure 3.2: Laminar-turbulent transition in the boundary layer on a flat plate at zero-pressure-gradient [82]

A principal mechanism influencing the laminar-turbulent transition process is the development and evolution of Tollmien-Schlichting (T-S) waves. T-S waves are streamwise instabilities which usually occur in the boundary layer. This complex phenomenon has been observed experimentally by Gaster, Kovasznay, Klebanoff, Schubauer and others, and for which theoretical predictions have been validated [83]. With reference to Figure 3.2, the fluid flows at freestream velocity along the flat plate until the critical Reynolds number, Re_{crit} is reached. As soon as the flow passes the Re_{crit}, it becomes unstable and Tollmien-Schlichting (T-S) waves become unstable. Further downstream, the T-S waves evolves as 3D perturbations and the flow starts showing span-wise variations, accompanied by vortices formation (Λ-vortices). As Re_x increases, these vortices break down to form turbulent spots through the formation of so-called hairpin vortices. During this stage of transition, the flow alternates between laminar and turbulent (intermittency effect). The hairpin vortices grow and merge together to form turbulent spots and eventually forming a fully turbulent boundary layer.

Laminar-turbulent transition is generally classified in different categories: natural, bypass, sepa-

rated flow, reverse and wake induced transition. Figure 3.3 illustrates the momentum Reynolds number versus the acceleration parameter plot with different transition categories [81]. The effect of freestream acceleration on the boundary layer is a key parameter in the classification of different transition types.

Figure 3.3: Momentum Re number vs acceleration for different types of transitional flows [81]

The acceleration parameter, K is defined as

$$K = \left(\frac{\nu}{U^2}\right)\left(\frac{dU}{dx}\right) \tag{3.2}$$

The acceleration parameter represents the effect of freestream acceleration on the boundary layer. Figure 3.3 can be utilised to decide/identify the type of transition [81]. Above the "stability criterion" line T-S waves develop. Above the "separation criterion" line, a laminar boundary layer separates, which can lead to separated flow transition. The horizontal shaded area correlates to the transitional Reynolds number (in the range of 5 to 10% turbulence level).

3.2.1.1 Natural transition

As described by Schlichting [82], natural transition is subject to the following primary stages: formation, amplification and breakdown. Natural transition phenomenon takes place when the instabilities migrating within the boundary layer are relatively small. The initiation of natural transition is primarily as a result of disturbances in the freestream region entering the boundary layer [84]. Natural transition is characterised by viscous Tollmien-Schlichting instability waves (the primary modes) which can be observed under quiet experimental testing conditions and can be calculated by employing linear stability theory [82]. The amplification of the T-S waves and the evolution towards formation of 3D perturbations are referred to as the secondary modes of natural transition. Natural transition is mainly observed at high Reynolds numbers (Re) low freestream turbulence levels, $Tu < 1.0\%$ [85]. As a result of the extremely low disturbance levels involved in natural transition, the boundary layer may remain laminar even when the Reynolds number increases well beyond that of by-pass transition. The Reynolds number at which natural transition occurs on a flat plate with constant pressure is approximately $Re \approx 5 \times 10^6$ [86]. The various stages of the natural transition process are explained in details in section 3.2.1, together with an illustrative representation in figure 3.2, with reference to flow on a flat plate at at zero-pressure-gradient.

3.2.1.2 By-pass transition

In a wide range of flows, the laminar to turbulent transition of boundary layers is driven by freestream perturbations, where transition takes place more rapidly, by-passing the natural route of transition (i.e. the first, second and third stages of natural transition), followed by the direct formation of turbulent spots in the boundary layer [81]. This transition mode is referred to as by-pass transition, and the term *by-pass transition* was first introduced in 1969 by Markovin [87], suggesting the possibility of by-passing the T-S wave route to transition, if substituted by an alternative mechanism which is highly amplifying so that the T-S wave route to transition could be by-passed.

By-pass transition is classed as the underlying mode of transition in many engineering applications where the FST levels are higher than the threshold level of approximately 0.5% [51]. Mayle [81] reported that for by-pass transition, the linear stability theory is invalid and that there has not been any reported evidence of T-S waves for FST greater than 1.0%. The FST value that is considered as the boundary between natural and by-pass transition has been quoted numerous times between 0.5% and 1.0% [51, 81]. The typical Reynolds number for a laminar boundary layer over a flat plate at constant pressure to undergo by-pass transition is $Re \approx 5 \times 10^5$ [86].

3.2.1.3 Separated flow transition

When a laminar boundary layer fails to overcome adverse pressure gradients, the boundary layer separates. Transition can be induced as a result of strong amplification of instabilities in the separated region. The boundary layer re-attaches as a result of the result turbulent flow enhancing mixing and momentum transfer in the direction normal to the wall. This process of laminar separation, transition and turbulent re-attachment is known as laminar separation bubbles (LSB) [88]. The stages involved in separated flow transition is highly dependent upon the magnitude of the adverse pressure gradient as well as the existence of other disturbances in the flow. This type of transition is very common near the leading edges of turbine blades and compressors, and aircraft wings, as well as Unmanned Aerial Vehicles.

Figure 3.4: Laminar separation bubbles on a NACA0012 aerofoil at 5 deg angle of attack [88]

The presence of LSB on an aerofoil has significant effect on the aerodynamic performance, typically increasing drag. Depending on their effect on the flow, the separation bubbles can be described as either "long" or "short" bubbles. A short separation bubble only affects a flow locally unlike a long separation bubble that may influence the overall pressure distribution [89]. The characterisation of the bubble length from short to long is known as bubble bursting. Bubble bursting takes place when a minor increment in the angle of incidence, that results in a rapid expansion in bubble length causing significant drop in aerodynamic performance. This is the primary reason of considering bubble bursting phenomenon in low Reynolds aerofoil design [88].

3.2.1.4 Reverse transition

This transition type, usually referred to as relaminarisation or retransition, is the process during which a turbulent flow transitions to laminar, under the influence of a strong favourable pressure gradient. Reverse transition is usually experienced as a result of the high accelerations on the high pressure side of aerofoils at the trailing edge and on the low pressure side of turbine aerofoils at the leading edge. Relaminarisation typically occurs when the acceleration parameter, K is greater than 3×10^6 [81].

3.2.1.5 Wake-induced transition

Wake-induced transition is an instance of by-pass transition which is usually experienced in turbomachinery flows [85] where the turbulence level of the background flow (usually around Tu = 10%) is elevated enough to force the production of turbulent spots in the boundary layer by overcoming the natural route to transition, which is as a result of turbulence fluctuations in the flow. Viscous wakes in turbomachinery flows initiate the turbulent spots formation, shifting the transition location [90].

The understanding of the general boundary layer transition phenomenon, together with the various transition types are very crucial, especially when developing new turbulence models with transition modelling capabilities. It is equally indispensable to identify the main factors influencing the transition process, which are described as follows.

3.2.2 Factors affecting transition

A major part that contributes towards the difficulty of fully understanding transition since it was first demonstrated by Osborne Reynolds in the late nineteenth century lies in the various factors affecting transition. These factors, that are in one way or another associated with each other, include freestream turbulence, pressure gradient, surface roughness, curvature, flow convergence or divergence, compressibility, heat transfer and film cooling. The presence of one or more of these factors at the same instance renders their individual effects to be non-additive. Progress in transition is ongoing towards the determination of the respective trends of these influential factors, that are indispensable to either explain or predict the transition phenomenon [91].

The onset of transition is primarily governed by freestream turbulence and unsteadiness [92]. Key factors such as pressure gradient, turbulence and the occurrence of flow separation have a significant effect on the production of turbulent spots. The other remaining factors do not have a major influence on the production of turbulent spots, their effect are in fact about 5-10 times less than that of pressure gradient.

The freestream turbulence affects the heat transfer rate in a turbulent boundary layer. Increasing the freestream turbulence increases the heat transfer and reduces the transition onset Reynolds number. This increases the production of turbulent spots and therefore reduces the transition length. Figure 3.5 provides a graphical representation of the turbulent spots production rate as a function of FST level (zero-pressure-gradient). At more elevated FST levels ($Tu > 1.0\%$), transition mode switches to by-pass transition, overpassing the natural route of transition.

Figure 3.5: Turbulent spots production rate vs turbulence level [81]

Transition onset and transition length are significantly affected by pressure gradient. In a favourable pressure gradient, transition onset takes place further downstream as a result of the stabilising effect on the flow, and therefore increasing the transition length. However, in an adverse pressure gradient, the flow is destabilised, promoting transition onset more upstream, decreasing the transition length. As discussed earlier in this Chapter, at very large adverse pressure gradient, the boundary layer separates. The acceleration parameter, K is considered as a correlative measurement of pressure gradient for flows at favourable pressure gradients. Figure 3.6 illustrates the relation between transitional Reynolds number and the acceleration parameter K.

Figure 3.6: Transitional Reynolds number vs acceleration parameter [81]

With reference to figure 3.6, the transitional Reynolds number increases as the acceleration parameter increases, which as a result delays transition. The effect of acceleration is critically large at low FST levels, unlike for flows at high FST levels (for instance, flows in gas turbines), the effect of acceleration is significantly low that it is considered to be negligible, since transition onset is controlled by the FST level [92]. Mayle [81] reported that the effect of FST level is much less substantial for flows with adverse pressure gradients compared to flows with favourable pressure gradients.

Surface roughness is considered to have a substantial effect on laminar-turbulent transition, promoting premature transition within the boundary layer. Surface roughness tend to cause the boundary layer to transition from laminar to turbulent in a shorter distance along the surface com-

pared to flow over a surface where the roughness is assumed to be negligible [80]. The negative influence that surface roughness has on transition has in turn affected the manufacturing cost of a wide range of engineering products, particularly turbine and aircraft engine blades. Roelke and Haas [93] carried out an experimental investigation to determine the influence of blade profile inaccuracies and surface finish on the aerodynamic performance of turbine blades. The original cast rotor blades had a considerably rough surface finish. The experimental test results, compared with the original blade, confirmed an increase in efficiency by an order of one when smoothing the surface finish of the blade.

A more detailed account on the previously discussed factors within the current work together with other factors influencing transition can be consulted from the following literature sources [81, 91, 92].

3.3 Turbulence & transition modelling

This section aims to provide a detailed account of the principal attributes of turbulence modelling, beginning with the description of the governing equations of any fluid flow problem, i.e. the Navier-Stokes equations, followed by a brief overview of the Boussinesq approximation and near-wall treatment of turbulence. The RANS and LES modelling approaches are presented as an understanding of these techniques is required in order to develop a Hybrid RANS/LES model. Additionally, a selection of turbulence models are detailed, including descriptions of the model equations (these models are assessed in Chapter 5 and Chapter 6).

3.3.1 Navier-Stokes equations

The Navier-Stokes equations have been recognised as the governing equations for laminar, transitional and turbulent flows for the past two centuries. These governing equations are a combination of non-linear partial differential equations, that describes any fluid motion in space and time.

Applying the principle of mass conservation, the continuity equation is obtained, which can bbe expressed (in Cartesian form) as follows

$$\underbrace{\frac{\partial \rho}{\partial t}}_{\text{temporal}} + \underbrace{\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z}}_{\text{spatial}} = 0 \tag{3.3}$$

Applying the principle of momentum conservation, the momentum equations are obtained, describing the motion of fluid through space, together with any force acting on the fluid. The momentum equations (in Cartesian form) are given by

$$\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}\right) = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + \rho g_x \tag{3.4}$$

$$\rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z}\right) = -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + \rho g_y \tag{3.5}$$

$$\rho\left(\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z}\right) = -\frac{\partial p}{\partial z} + \mu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + \rho g_z \tag{3.6}$$

The continuity and momentum equations can be re-written in a general differential form (not considering the coordinates) as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{3.7}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \sum \mathbf{f} \tag{3.8}$$

For Newtonian fluids, where there is a linear relation between the velocity gradient and shear stress, the momentum equation (3.8) can be re-written as follows, neglecting the forces due to gravity

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \nabla \cdot \mathbf{T} \tag{3.9}$$

where $\mathbf{T}$ is the stress tensor and is defined as

$$\mathbf{T} = -\left(p + \frac{2}{3}\mu \nabla \cdot \mathbf{u}\right)\mathbf{I} + \mu\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right) \tag{3.10}$$

For the detailed derivation of the Navier-Stokes equations, standard literature [94, 95] can be referred to.

For incompressible fluid flow, the density is constant. The viscosity can also be assumed to be constant under isothermal conditions. Hence, equations (3.7) and (3.9) can be simplified to

$$\nabla \cdot \mathbf{u} = 0 \tag{3.11}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} \tag{3.12}$$

3.3.2 The Boussinesq hypothesis

The Boussinesq hypothesis [96, 97] was proposed in 1877 by Joseph Valentin Boussinesq, a French mathematician and physicist. Boussinesq postulated that the momentum transfer as a result of turbulent eddies can be modelled with an eddy viscosity. The Boussinesq assumption states that the Reynolds stress tensor, τ_{ij} is directly proportional to the mean strain rate tensor, S^*_{ij}, given by

$$\tau_{ij} = 2\mu_t S^*_{ij} - \frac{2}{3}\rho k \delta_{ij} \tag{3.13}$$

μ_t is a scalar property called the eddy viscosity. The eddy viscosity is a flow property rather than a fluid property (unlike the molecular viscosity), and hence is influenced by the flow characteristics.

Equation (3.13) can be explicitly re-written as

$$-\rho \overline{u_i' u_j'} = \mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho k \delta_{ij} \tag{3.14}$$

It is to be noted that for incompressible flows,

$$\frac{\partial U_k}{\partial x_k} = 0 \tag{3.15}$$

The Boussinesq hypothesis, also referred to as the Boussinesq eddy viscosity approximation, simplifies the treatment of turbulence since the Reynolds stresses are related to the mean flow. A range of turbulence models are developed based on the Boussinesq Hypothesis that can be characterised based on the number of additional transport equations used together with the momentum equations for the eddy viscosity computation [98]. These turbulence models are generally of three types: zero-equation models, one-equation models, and two-equation models.

3.3.3 Wall bounded turbulence

The properties of the different scales in turbulent flow have critical influence when it comes to the treatment of turbulence. It is also important to highlight the effects of solid boundaries on fluid flow. The larger energetic scales in turbulent flow are severely influenced by the case geometry or the flow topology. For the majority of the engineering applications, the distance to the wall is considered as a crucial parameter, which is responsible for the occurrence of anisotropy in turbulence. Away from the wall, turbulence is relatively isotropic, transitioning to anisotropic while approaching closer to the wall. When the fluid is in direct contact with the wall, the velocity is zero, i.e. the "no-slip" condition applies, $u_w = v_w = w_w = 0$. At the wall, the turbulent motions closest to the wall are totally eliminated as a result of the increased damping effect on the turbulent fluctuations normal to the wall. This implies that at the wall, the effect of the turbulent viscosity is negligible. This is usually implemented in turbulence models as the viscous damping function, f_ν.

Boundary layer velocity profiles for various types of flows are remarkably different with each flow type exhibiting incomparably non-linear profiles. However, the innovative effort of Ludwig Prandtl [99] and Theodore von Kármán [100] introduced the sub-division of the boundary layers in general regions, postulating the "law of the wall".

3.3.3.1 Law of the wall

The law of the wall attempts to renormalise the flow parameters as well as the geometric parameters in terms of the wall shear stress, τ_w. The renormalised parameters (scaled parameters) are the wall friction units and are represented by the superscript $^+$.

Figure 3.7 illustrates the wall regions and layers defined in terms of y^+ and y/δ.

Figure 3.7: Wall regions and layers defined in terms of y^+ and y/δ [95]

The subdivision of the boundary layer in various regions made non-dimensional description of boundary layer flows possible, which in turn allowed easy comparison between flows that are significantly different. Since non-dimensional velocity profiles can be obtained, the representation

of flow properties using wall coordinates is feasible. A detailed description of the various stages involved towards deriving the wall friction units representation of the flow parameters is as follows.

The local viscous time scale is given by

$$t_\nu = \left(\frac{\partial \overline{u}}{\partial y}|_{y=0} \right)^{-1} \tag{3.16}$$

$\overline{u}$ is the mean velocity in the streamwise direction, y is the coordinate normal to the wall.

At the wall [95], the effect of Reynolds stresses are negligible and the wall shear stress, τ_w can be mathematically represented by

$$\tau_w \equiv \mu \left(\frac{\partial \overline{u}}{\partial y} \right)_{y=0} \tag{3.17}$$

Substituting equation (3.17) into equation (3.16) gives

$$t_\nu = \frac{\mu}{\tau_w} \tag{3.18}$$

the viscous length scale can be defined as

$$l_\nu = \sqrt{\nu t_\nu} \tag{3.19}$$

The wall friction velocity, u_τ can be written as

$$u_\tau = \frac{l_\nu}{t_\nu} = \sqrt{\frac{\tau_w}{\rho}} \tag{3.20}$$

Expressing the wall-normal distance and the velocity in terms of wall friction units

$$y^+ = \frac{u_\tau y}{\nu} \tag{3.21}$$

$$u^+ = \frac{u}{u_\tau} \tag{3.22}$$

Eventually, the wall friction Reynolds number, Re_τ can be derived

$$Re_\tau = \frac{u_\tau \delta}{\nu}$$

where δ is the boundary layer thickness.

The regions, as illustrated in figure 3.7, are characterised by different physical properties. The different wall regions and layers, their individual locations in terms of wall coordinates and their physical properties are described in table 3.1.

Table 3.1: Wall regions and layers and their defining properties [95]

Region	Location	Defining Property
Inner layer	$y/\delta < 0.1$	The mean velocity profile is determined by u_τ and y^+, independent of U_0 and δ
Viscous wall region	$y^+ < 50$	The viscous contribution to the shear stress is significant, i.e. high viscosity effect
Viscous sublayer	$y^+ < 5$	The Reynolds shear stress is negligible when compared to the viscous shear stress, $u^+ = y^+$ y^+ is dimensionless wall distance y normal to the wall and u^+ is the dimensionless velocity
Outer layer	$y^+ > 50$	The direct effect of viscosity on the mean velocity profile is negligible
Overlap region	$y^+ > 50$, $y/\delta < 0.1$	The region of overlap between inner and outer layers (only occurs at high Reynolds numbers)
Log-law region	$y^+ > 30$, $y/\delta < 0.3$	The region where the log-law applies, $u^+ = \frac{1}{\kappa} \ln y^+ + B$ $\kappa = 0.41$ and $B = 5.2$
Buffer layer	$5 < y^+ < 30$	The transitional region between the viscous sublayer and the log-law region $u^+ \neq y^+$ and $u^+ \neq \frac{1}{\kappa} \ln\ y^+ + B$ neither of the above laws apply

3.3.3.2 Near-wall treatment

As stated earlier, the presence of a solid boundary has significant effects on the computation of turbulent flows. The two main effects of the presence of a wall that are considered to be imperative in fluid flow computations are the transition of turbulent flow from isotropic (away from the wall) to anisotropic (close to the wall) and the increased turbulence production as a result of the shear mechanism. These wall effects on the flow have a major influence on turbulence modelling. When a turbulence model is not capable to reproduce the near-wall characteristics, either a damping function or a wall function is employed.

Damping functions are used by low Reynolds number turbulence models, which involves actual resolution of the region close to the wall proximity. This approach of damping function requires a density intensive grid ($y^+ \leqslant 1$) in order to resolve the turbulent scales in the viscous sublayer appropriately and hence increasing the computational cost. Figure 3.8 illustrates the viscous sublayer modelling using fine grids near the wall.

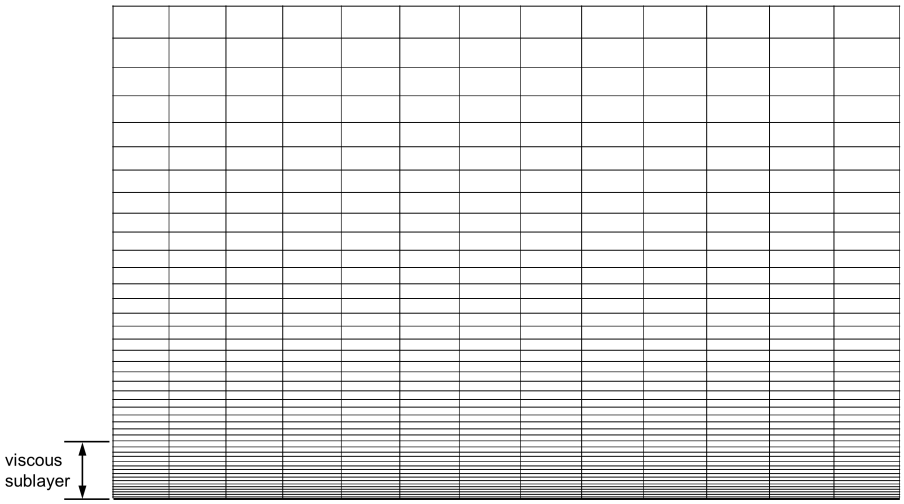


Figure 3.8: Viscous sublayer modelling using fine grids near the wall

The other approach to modelling near-wall turbulence is to treat the near-wall flow with a wall function. Wall functions are usually employed by high Reynolds number turbulence models. Wall functions are simply semi-empirical mathematical formulae characterising the behaviour of the flow in the viscous sublayer under standard conditions. A key advantage of wall functions is the significant reduction in computational cost as a result of the use of lower grid density ($30 \leqslant y^+ \leqslant 100$) than damping function approach. A graphical representation of the application of wall function to the viscous sublayer is illustrated in figure 3.9.

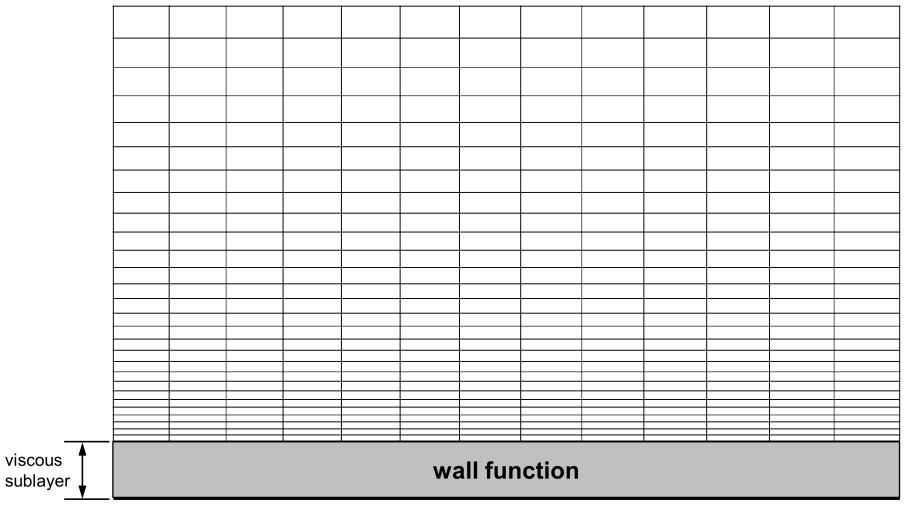


Figure 3.9: Viscous sublayer modelling using wall function

When using wall functions, the computational grid has to be carefully designed in order to avoid interference with the wall function. The grid at the wall has to be coarse enough ($y^+ \geqslant 30$) so that the first cell from the wall lies in the log-layer, where the production and dissipation of the turbulent kinetic energy are in equilibrium thereby reducing turbulent instability in the near wall computation. This will therefore avoid resolving the viscous sublayer. The performance of wall functions is affected when applied to complex geometries with strong curvatures, where large flow separations occur. Despite their limitations, wall functions are widely used in CFD modelling, particularly in industry, due to their reduced computational requirement, in comparison to the viscous sublayer modelling method.

3.3.4 Reynolds-averaged Navier-Stokes (RANS)

3.3.4.1 Governing equations

The incompressible fluid motion is governed by the continuity equation and the momentum equation. The continuity equation (3.11) and the momentum equation (3.12) can be re-written in Einstein notation as

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3.23}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu\frac{\partial^2 u_i}{\partial x_i \partial x_j} \tag{3.24}$$

In turbulent flows, the flow field properties become random spatial and temporal functions. The flow field variables, u_i and p are separated into the mean and fluctuating part. Hence, the decomposition of velocity and pressure, in terms of the sum of mean and fluctuating parts, with the bar denoting time-average, can be written as

$$u_i = \overline{u}_i + u_i' \qquad p = \overline{p}_i + p' \tag{3.25}$$

Substituting (3.25) into (3.23) and (3.24) and simplifying further to take into account the Reynolds stress term, gives the Reynolds-averaged Navier-Stokes equations as illustrated below

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0 \tag{3.26}$$

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu\frac{\partial^2 \overline{u}_i}{\partial x_i \partial x_j} - \frac{\partial \overline{u_i' u_j'}}{\partial x_j} \tag{3.27}$$

Equations (3.26) and (3.27) above are in the same form of the Navier-Stokes equation, with the exception of the term $\overline{u_i' u_j'}$, usually referred to as the Reynolds stress tensor but is in fact representing the momentum flux as a result of the turbulent fluctuations flowing in an out of a control volume (CV). The decomposition of the flow properties into the mean and the fluctuating parts produced three unknowns, which cannot be solved since additional equations are required to close the system. Hence, RANS based models are employed for the system closure and provide numerical solutions.

RANS turbulence models are generally classified into two main categories with respect to the near-wall treatment of turbulence: high and low Reynolds number models. As part of the methodology of this current work, testing a range of both high and low Reynolds number turbulence models for their capability to predict transition is a crucial practice, which will be described later in Chapter 5. These turbulence models are described in details, highlighting the different mathematical terms and equations in Appendix A

3.3.4.2 High Reynolds number models

High Reynolds number turbulence models are designed to solve complex cases, for which achieving a detailed grid resolution at the wall is a challenge. Therefore, the use of wall functions at no-slip walls is crucial in order to solve the wall grid resolution problem. In terms of modelling the boundary layer region, high Re models have modelling capabilities that would describe to certain accuracy, the nature of the flow, therefore it is possible to have a y^+ value between 30 and 100, giving it the advantage, over low Re models, of using less computational resources.

The following high Reynolds number models were tested as part of the current work: $k-\epsilon$ model, $k-\omega$ model and $k-\omega$ SST model. A detailed description of these high Reynolds number models is provided in Appendix A.

3.3.4.3 Low Reynolds number models

The low Reynolds number turbulence models are designed to resolve the flow down to the viscosity dominated sublayer and wall function approaches which close the gap between a node within a turbulent region and the wall with a well-defined profile. This type of turbulence models can be said to integrate right to the wall, giving a more detailed image of what is actually happening in the simulation. These types of simulations require the grid to be well defined at the wall ($y^+ \sim 1$), ensuring that there is enough resolution in the viscous sublayer to capture near-wall behaviour.

The following low Reynolds number models were tested as part of the current work: Launder-Sharma $k-\epsilon$ model, v^2-f model and $q-\zeta$ model. A detailed description of these low Reynolds number models is provided in Appendix A.

3.3.4.4 $k_T - k_L - \omega$ transition model

The $k_T - k_L - \omega$ model is a three equation eddy-viscosity model that employs three additional transport equations. This model, developed by Walters and Cokljat [2] is based on the $k-\omega$ turbulence model, with a key feature of modelling the low frequency pre-transitional fluctuations, known as the laminar kinetic energy and is denoted by k_L, which is essentially the energy of streamwise fluctuations in pre-transitional region. These streamwise fluctuations under further development leads to by-pass transition, which is a key feature of boundary layer transition. The $k_T-k_L-\omega$ model of Walters and Cokljat has been developed to model these transitional features. The $k_T - k_L - \omega$ model described is the corrected version presented by Medina and Early [7], which solves the problem leading to strong underestimation of friction in the turbulent region.

The transport equations of the three-equation eddy viscosity model of Walters and Cokljat [2], including the turbulent kinetic energy, k_T, the laminar kinetic energy, k_L and the specific dissipation rate, ω are

$$\frac{Dk_T}{Dt} = P_{k_T} + R_{BP} + R_{NAT} - \omega k_T - D_T + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\alpha_T}{\sigma_k}\right)\frac{\partial k_T}{\partial dx_j}\right] \tag{3.28}$$

$$\frac{Dk_L}{Dt} = P_{k_L} - R_{BP} - R_{NAT} - D_L + \frac{\partial}{\partial x_j}\left[\nu\frac{\partial k_L}{\partial dx_j}\right] \tag{3.29}$$

$$\frac{D\omega}{Dt} = C_{\omega_1}\frac{\omega}{k_T}P_{k_T} + \left(\frac{C_{\omega R}}{f_W} - 1\right)\frac{\omega}{k_T}(R_{BP} + R_{NAT}) - C_{\omega 2}\omega^2 + C_{\omega 3}f_\omega\alpha_T f_W^2\frac{\sqrt{k_T}}{d^3} + \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\rho\alpha_T}{\alpha_\omega}\right)\frac{\partial\omega}{\partial x_j}\right] \tag{3.30}$$

The various terms in the equations (3.59), (3.60) and (3.61) represent advection, production, destruction and diffusion.

The production of turbulent and laminar kinetic energy is

$$P_{k_T} = \nu_{T,s}S^2 \tag{3.31}$$

$$P_{k_L} = \nu_{T,l}S^2 \tag{3.32}$$

where $S = \sqrt{2S_{ij}S_{ij}}$ is the absolute value of strain.

$\nu_{T,s}$ and $\nu_{T,l}$ represent the small- and large-scale eddy viscosity respectively. The small-scale eddy viscosity is defined as

$$\nu_{T,s} = f_\nu f_{INT} C_\mu \sqrt{k_{T,s}} \lambda_{eff} \tag{3.33}$$

where $k_{T,s}$ is the effective small-scale turbulence.

$$k_{T,s} = f_{SS} f_W k_T \tag{3.34}$$

The kinematic wall effect is modelled using a wall limited turbulence length scale λ_{eff} and damping function f_W is given by

$$\lambda_{eff} = \min\left(C_\lambda d, \lambda_T\right) \tag{3.35}$$

where

$$\lambda_T = \frac{\sqrt{k_T}}{\omega} \tag{3.36}$$

The damping function is given by

$$f_W = \left(\frac{\lambda_{eff}}{\lambda_T}\right)^{\frac{2}{3}} \tag{3.37}$$

where d is the wall distance.

The damping function represented by equation (3.68) is different from the work presented by Walters and Cokljat [2]. The exponent $2/3$ was introduced so as to maintain consistency between the various terms in the equations of this model and the previous transition model developed by Walters et al. [39].

In order to model the viscous wall effects, a viscous damping function f_ν is used, which is dependent on the effective turbulence Reynolds number, Re_T

$$f_\nu = 1 - \exp\left(-\frac{\sqrt{Re_T}}{A_\nu}\right) \tag{3.38}$$

$$Re_T = \frac{f_W^2 k_T}{\nu\omega} \tag{3.39}$$

The shear sheltering effect is incorporated through the damping function

$$f_{ss} = \exp\left[-\left(\frac{C_{ss}\nu\Omega}{k_T}\right)^2\right] \tag{3.40}$$

The turbulent viscosity coefficient, C_μ becomes

$$C_\mu = \frac{1}{A_0 + A_s\left(\frac{S}{\omega}\right)} \tag{3.41}$$

The intermittency effect on the turbulence production is described using an intermittency damping function

$$f_{INT} = \min\left(\frac{k_T}{C_{INT}\left(k_T + k_L\right)}, 1\right) \tag{3.42}$$

The expression illustrated in equation (3.73) was corrected by Medina and Early [7]. Walters and Cokljat [2] described the intermittency damping function with k_L in the numerator.

The production of laminar kinetic energy, k_L is defined by the large-scale near wall turbulence

$$k_{T,l} = k_T - k_{T,s} \tag{3.43}$$

The production of laminar kinetic energy is modelled as a product of the large scale eddy viscosity and the square of the magnitude of the mean strain. The production term is given by

$$P_{k_L} = \nu_{T,l} S^2 \tag{3.44}$$

where

$$\nu_{T,l} = \min\{f_{\tau,l} C_{11} \left(\frac{\Omega \lambda_{eff}^2}{\nu}\right) \sqrt{k_{T,l}} \lambda_{eff} + \beta_{TS} C_{12} Re_\Omega d^2 \Omega, \frac{k_L - k_{T,l}}{2S}\} \tag{3.45}$$

$$Re_\Omega = \frac{d^2 \Omega}{\nu} \tag{3.46}$$

$$\beta_{TS} = 1 - \exp\left(-\frac{\max\left(Re_\Omega - C_{TS,crit}, 0\right)^2}{A_{TS}}\right) \tag{3.47}$$

$$f_{\tau,l} = 1 - \exp\left(-C_{\tau,l} \frac{k_{T,l}}{\lambda_{eff}^2 \Omega^2}\right) \tag{3.48}$$

The anisotropic dissipation terms which appear in the transport equations for k_L and k_T are modelled as shown below

$$D_T = \nu \frac{\partial \sqrt{k_T}}{\partial x_j} \frac{\partial \sqrt{k_T}}{\partial x_j} \tag{3.49}$$

$$D_L = \nu \frac{\partial \sqrt{k_L}}{\partial x_j} \frac{\partial \sqrt{k_L}}{\partial x_j} \tag{3.50}$$

The above anisotropic dissipation terms were proposed by Walters and Cokljat [2]. However, in previous models, for example, the Launder-Sharma $k-\epsilon$ model, the above anisotropic dissipation terms multiplied by a factor of 2. The newly proposed terms were used for computations [2,7,42], which produced good results.

The turbulent diffusivity, α_T is given by

$$\alpha_T = f_\nu C_{\mu,std} \sqrt{k_{T,s}} \lambda_{eff} \tag{3.51}$$

The damping function, f_ω is given by

$$f_\omega = 1 - \exp\left[-0.41 \left(\frac{\lambda_{eff}}{\lambda_T}\right)^4\right] \tag{3.52}$$

R_{BP} and R_{NAT} express the laminar-turbulent transition in terms of the energy transfer from k_L to k_T

$$R_{BP} = C_R \beta_{BP} k_L \omega / f_W \tag{3.53}$$

$$R_{NAT} = C_{R,NAT} \beta_{NAT} k_L \Omega \tag{3.54}$$

The bypass transition, when considered in modelling laminar-transition, is given by

$$\beta_{BP} = 1 - \exp\left(-\frac{\phi_{BP}}{A_{BP}}\right) \tag{3.55}$$

$$\phi_{BP} = \max\left[\left(\frac{k_T}{\nu\Omega} - C_{BP,crit}\right), 0\right] \tag{3.56}$$

β_{BP} in the above expression acts as a threshold function, initiating the transition from laminar to turbulent flow when bypass transition is taken into account. β_{BP} is controlled by including the limiting function ϕ_{BP}. This term was modified since the original form as illustrated in Walters and Cokljat (2008) paper leads to a non-physical response for the estimated transition locations at low turbulence intensities (Tu<0.9%).

The natural transition, when considered in modelling laminar-turbulent transition, is given by

$$\beta_{NAT} = 1 - \exp\left(-\frac{\phi_{NAT}}{A_{NAT}}\right) \tag{3.57}$$

$$\phi_{NAT} = \max\left[\left(Re_{\Omega}\frac{C_{NAT,crit}}{f_{NAT,crit}}\right), 0\right] \tag{3.58}$$

$$f_{NAT,crit} = 1 - \exp\left(-C_{NC}\frac{\sqrt{k_L}d}{\nu}\right) \tag{3.59}$$

The turbulent kinematic viscosity used in the momentum equations can be written as

$$\nu_T = \nu_{T,s} + \nu_{T,l} \tag{3.60}$$

The turbulent thermal diffusivity α_θ is defined as

$$\alpha_\theta = f_W\frac{k_T}{k_T + k_L}\frac{\nu_T, s}{Pr_\theta} + (1 - f_W)\, C_{\alpha,\theta}\sqrt{k_T}\lambda_{eff} \tag{3.61}$$

The coefficient $C_{\omega 2}$ is given by

$$C_{\omega 2} = 0.92 f_W^2 \tag{3.62}$$

The boundary conditions at no-slip wall are $k_T = k_L = 0$ and a zero-gradient for ω

$$\frac{\partial\omega}{\partial\eta} = 0 \tag{3.63}$$

The model constants are illustrated in table 3.2 below.

Table 3.2: k_T k_L ω transition model constants [2]

3.3.5 Large-eddy simulation (LES)

3.3.5.1 Governing equations

As briefly discussed in Chapter 2, LES operates on the basis of resolving the large scales of turbulence and modelling the small scales. In order to separate the small scales [101] from the large scales, some form of averaging must be carried out which is not at all similar to the ensemble averaging process of the RANS approach. The LES approach employs the definition of a filtering operation, which is a locally established weighted average of flow properties with respect to a fluid volume and is applied to the governing equations of fluid motion, the Navier-Stokes equations. Based on the principal of LES, for any flow parameter, f, there exist a contribution of a large scale and a small scale, which can be mathematically represented by

$$\overline{f} = f - f' \tag{3.64}$$

Thereafter, the overbar and the prime components represent the large scales and the small scales respectively. The LES filtering operation can be defined as follows:

$$\overline{f}(x) = \oint G\left(x, x'; \Delta\right) f(x')\, \mathrm{d}x' \tag{3.65}$$

where Δ is the filter width and is also proportional to the smallest scale wavelength due to the filtering operation. The filter kernel, $G(x, x'; \Delta)$ is a localised function that must satisfy the following condition,

$$\oint G\left(x, x'; \Delta\right) \mathrm{d}x' = 1 \tag{3.66}$$

A more in-depth description of the various types of LES filters can be accessed from the textbook by Segaut and Méneveau [102].

When the filtering operation is applied to the Navier-Stokes equations for incompressible flow, the filtered incompressible Navier-Stokes equations are obtained. The filtered continuity equation reads

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0 \tag{3.67}$$

The filtered momentum equation is given by

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \overline{u_i u_j}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_i \partial x_j} \tag{3.68}$$

which appears to be identical to equation (3.28) (ignoring the Reynolds stresses). However since,

$$\overline{u_i u_j} \neq \overline{u}_i \overline{u}_j \tag{3.69}$$

a modelling approximation has to be introduced to take into account the difference of the LHS term and the RHS term of equation (3.100). This is defined as

$$\tau_{ij} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j \tag{3.70}$$

Therefore, equation (3.99) can be re-written

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} - \frac{\partial \tau_{ij}}{\partial x_j} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_i \partial x_j} \tag{3.71}$$

In the context of LES, the term τ_{ij} is referred to as the subgrid-scale stress tensor and it accounts for the the effect of small unresolved scales, which are required to be modelled.

The subgrid-scale stress tensor, τ_{ij} can be decomposed into other terms by decomposing the velocity field as $u = \overline{u} + u'$, yielding

$$\tau_{ij} = \overline{\left(\overline{u}_i + u'_i\right)\left(\overline{u}_j + u'_j\right)} - \overline{u}_i\overline{u}_j = \underbrace{\left(\overline{\overline{u}_i\overline{u}_j} - \overline{u}_i\overline{u}_j\right)}_{\substack{\text{Leonard}\\ \text{stress}}} + \underbrace{\left(\overline{\overline{u}_i u'_j} + \overline{u'_j\overline{u}_i}\right)}_{\substack{\text{cross-term}\\ \text{stress}}} + \underbrace{\overline{u'_i u'_j}}_{\substack{\text{SGS}\\ \text{Reynolds}\\ \text{stress}}} \tag{3.72}$$

A detailed account on the various stresses as represented by equation (3.103) can be accessed from [101].

3.3.5.2 Smagorinsky model

The Smagorinsky model [47] is the very first subgrid-scale model introduced in the early 1960s by the founder of the LES approach, Joseph Smagorinsky. There has been a wide application of this SGS model since its introduction and it is still one of the most used SGS models. The Samgorinsky model is a zero-equation eddy-viscosity model that functions based upon the Boussinesq approximation to model the subgrid-scale stresses. The subgrid stress tensor τ_{ij} can be written as

$$\tau_{ij} = \frac{1}{3}\tau_{kk}\delta_{ij} - 2\left(C_s\Delta\right)^2\left|\overline{S}\right|\overline{S}_{ij} \tag{3.73}$$

The term $\overline{S}_{ij}$ is referred to as the strain rate tensor, which is computed with the filtered velocity $\overline{u}$ and the expression is given by

$$\overline{S}_{ij} = \frac{1}{2}\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right) \tag{3.74}$$

The eddy viscosity, as a function of the strain rate tensor is given by

$$\nu_{sgs} = \left(C_S\Delta\right)^2\left|\overline{S}\right| \tag{3.75}$$

where

$$\overline{S} = \sqrt{2\overline{S}_{ij}\overline{S}_{ij}} \tag{3.76}$$

and C_S is the Smagorinsky constant. The value of C_S varies for various types of flows. Lilly [103] assigned a values for C_S lying in the range of 0.18 to 0.23. The channel flow simulation by Deardorff [48] established a relatively lower value, which as been cited in numerous occasions as 0.08 to 0.11.

Δ is the filter width, which is given by

$$\Delta = \left(\Delta_x\Delta_y\Delta_z\right)^{\frac{1}{3}} \tag{3.77}$$

It is to be noted that when applying the stress tensor, τ_{kk} to the pressure, the filtered momentum equation can be re-written as

$$\frac{\partial\overline{u}_i}{\partial t} + \overline{u}_j\frac{\partial\overline{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial\overline{q}}{\partial x_i} - \frac{\partial\tau_{ij}}{\partial x_j} + \nu\frac{\partial^2\overline{u}_i}{\partial x_i\partial x_j} \tag{3.78}$$

where

$$\overline{q} = \overline{p} - \frac{1}{3}\rho\delta_{ij}\tau_{kk} \tag{3.79}$$

As discussed in Chapter 2, the Smagorinsky model inherent several limitations. A main drawback is that the model fails to predict transition as a result of the positive and uniform Smagorinsky constant, C_S, which produces eddy viscosity in the sheared laminar flow region [104]. Moreover, the model struggles to replicate the near-wall effect in the viscous sublayer, due to which the model requires a damping function to alleviate this problem to some extent. One of the common drawbacks of the Smagorinsky model is that it is highly dissipative, i.e. the invariable extraction of energy from the resolved scales by the subgrid-scales [104].

3.3.5.3 One equation eddy viscosity model

In order to improve the accuracy issue faced by the Smagorinsky model, transport equation subgrid-scale models were introduced, aiming towards discarding the assumption of the existence of equilibrium between the production and destruction of subgrid energy from the unresolved scales. Several of these model types have been introduced, but a common example is the one eqaution eddy viscosity model of Yoshizawa [105], which is very similar to the one implemented in OpenFOAM. This model consists of the SGS turbulent kinetic energy, k_{sgs} equation as the transport equation as is given by

$$\frac{\partial k_{sgs}}{\partial t} + \overline{u}_j \frac{\partial k_{sgs}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_{sgs}}{\sigma_{sgs}}\right)\frac{k_{sgs}}{\partial x_j}\right] + P - \frac{1}{\Delta} C_\epsilon k^{\frac{3}{2}} \tag{3.80}$$

where

$$P = \nu_{sgs}\overline{S}^2 \tag{3.81}$$

and

$$\overline{S} = \sqrt{2\overline{S}_{ij}\overline{S}_{ij}} \tag{3.82}$$

The eddy viscosity is is given by

$$\nu_{sgs} = C_k \Delta \sqrt{k_{sgs}} \tag{3.83}$$

The values for the constants $C_k = 0.094$ and $C_\epsilon = 1.048$. These are the default values as assigned in OpenFOAM. $\sigma_{sgs} = 1.0$.

Generally, the one equation eddy viscosity model have pretty much the same limitation as the Smagorinsky model as a result of employing the eddy viscosity concept and the assumption of equilibrium of the unresolved scales. However, due to the fact that the one equation eddy viscosity model additionally has a transport equation through which the velocity scale is independently defined, the time scale is more accurate. Fureby et al. [106] carried out a performance study of various SGS models and the results indicated that the one equation model performed better than the Smagorinsky type models.

3.3.5.4 Dynamic one equation eddy viscosity model

Before the dynamic one equation one equation eddy viscosity model is described, it is of paramount importance to highlight the procedure behind the formulation of dynamic type SGS models.

Germano's Dynamic Procedure

Germano et al. [52] proposed the dynamic procedure towards alleviating the problem faced by the Smagorinsky type models. In the Germano's dynamic procedure together with the modifications proposed by Lilly [107], the model constant, C (C_S in the case of the Smagorinsky model),

referred to as C_* (throughout the dynamic procedure), is determined dynamically and locally in space and time. C_* is computed at every time step as a function of the location, from details already present within the resolved velocity field. This is achieved by using a test filter $\widehat{\Delta}$ larger than the grid scale filter width Δ (2Δ).

As represented by equation (3.101), when applying the LES filtering operation, the subgrid stress tensor τ_{ij} was given as

$$\tau_{ij} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j \tag{3.84}$$

which was expressed as equation (3.104) for the Smagorinsky model. This equation was re-written by Piomelli and Liu [108], considering the model coefficient which is to be determined dynamically, as

$$\tau_{ij} = \frac{1}{3}\tau_{kk}\delta_{ij} - 2C_*\Delta^2 \left|\overline{S}\right| \overline{S}_{ij} \tag{3.85}$$

where C_* substituted the value C_S^2 of the Smagorinsky model.

Following the same principle, the application of the new test filter yields the subtest scale (STS) stress, T_{ij} as is given by

$$T_{ij} = \frac{1}{3}T_{kk}\delta_{ij} - 2C_*\widehat{\Delta}^2 \widehat{\left|\overline{S}\right|}\widehat{\overline{S}_{ij}} \tag{3.86}$$

where $\widehat{\Delta}$ is the test filter and $\widehat{\Delta} = 2\Delta$

and

$$\widehat{\overline{S}_{ij}} = \frac{1}{2}\left(\frac{\partial \widehat{\overline{u}_i}}{\partial x_j} + \frac{\partial \widehat{\overline{u}_j}}{\partial x_i}\right) \tag{3.87}$$

The revolutionary contribution of Germano et al. [52] in the SGS modelling problem is to recognise that a proper local choice of C_* is very important in order to maintain consistency between equations (3.116) and (3.117). As described in [107], the proper local choice of C_* can be fulfilled from the difference between T_{ij} and $\widehat{\tau}_{ij}$ given by

$$L_{ij} = T_{ij} - \widehat{\tau}_{ij} = \widehat{\overline{u}_i \overline{u}_j} - \widehat{\overline{u}}_i \widehat{\overline{u}}_j \tag{3.88}$$

Substituting equations (3.116) and (3.117) into (3.119) gives

$$L_{ij} = -2C_*\widehat{\Delta}^2 \widehat{\left|\overline{S}\right|}\widehat{\overline{S}_{ij}} + 2C_*\Delta^2 \widehat{\left|\overline{S}\right| \overline{S}_{ij}} \tag{3.89}$$

$$L_{ij} = 2C_* M_{ij} \tag{3.90}$$

Since equation (3.120) represents 5 independent equations that cannot be solved explicitly, the least square approach, which is a modification of the Germano subgrid-scale closure method, proposed by Lilly [107], is applied. Q is defined as the square of the error in equation (3.120)

$$Q = \left(L_{ij} + 2C_*\widehat{\Delta}^2 \widehat{\left|\overline{S}\right|}\widehat{\overline{S}_{ij}} - 2C_*\Delta^2 \widehat{\left|\overline{S}\right| \overline{S}_{ij}}\right)^2 \tag{3.91}$$

$$Q = \left(L_{ij} - 2C_* M_{ij}\right)^2 \tag{3.92}$$

Setting $\partial Q/\partial C_* = 0$, C_* can be evaluated as

$$C_* = -\frac{1}{2}\left(\frac{L_{ij}M_{ij}}{M_{ij}M_{ij}}\right) \tag{3.93}$$

The Germano dynamic procedure has been proven to overcome the limitations of the Smagorinsky type models. The dynamically computed value of C_* has shown to be yield accurate results and a range of flow types. Additionally, the dynamic procedure ensures that $C_* = 0$ in regions where the flow is laminar, allowing prediction of laminar-turbulent transition.

Ghosal et al. [109] proposed a dynamic localisation model one equation, that has some similarity to the dynamic model implemented in OpenFOAM. Since the reference to the model implementation in OpenFOAM was not available, analysing the source code, it was revealed that the dynamic one equation eddy viscosity model is simply an application of the dynamic procedure to the one equation eddy viscosity model Yoshizawa [105]. The model is described as follows.

The SGS kinetic energy equation is given by

$$\frac{\partial k_{sgs}}{\partial t} + \overline{u}_j \frac{\partial k_{sgs}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_{sgs}}{\sigma_{sgs}}\right)\frac{k_{sgs}}{\partial x_j}\right] + P - \frac{1}{\Delta} C_* k^{\frac{3}{2}} \tag{3.94}$$

where

$$P = \nu_{sgs}\overline{S}^2 \tag{3.95}$$

and

$$\overline{S} = \sqrt{2\overline{S}_{ij}\overline{S}_{ij}} \tag{3.96}$$

The eddy viscosity is is given by

$$\nu_{sgs} = C_{k_{dyn}} \Delta \sqrt{k_{sgs}} \tag{3.97}$$

$C_{k_{dyn}}$ and C_* are determined dynamically as described by the equations below

$$C_{k_{dyn}} = \frac{1}{2}\frac{L_{ij}\sigma_{ij}}{\sigma_{ij}\sigma_{ij}} \tag{3.98}$$

$$\sigma_{ij} = -\widehat{\Delta}\sqrt{k_{test}}\,\widehat{\overline{S}_{ij}} \tag{3.99}$$

$$L_{ij} = \widehat{\overline{u}_i\overline{u}_j} - \widehat{\overline{u}}_i\widehat{\overline{u}}_j \tag{3.100}$$

$$k_{test} = \frac{1}{2}L_{ij} \tag{3.101}$$

$$C_* = \frac{\widehat{\Delta}}{k_{test}^{\frac{3}{2}}}\left(\nu + \nu_{sgs}\right)\left[\widehat{\frac{\partial\overline{u}_i}{\partial x_j}\frac{\partial\overline{u}_i}{\partial x_j}} - \frac{\partial\widehat{\overline{u}}_i}{\partial x_j}\frac{\partial\widehat{\overline{u}}_i}{\partial x_j}\right] \tag{3.102}$$

$$\tau_{ij} = -2C_k\Delta\sqrt{k_{sgs}}\overline{S}_{ij} \tag{3.103}$$

The physics of the boundary layer transition has been discussed and the RANS and LES models described will be tested on a 2D zero-pressure-gradient flat plate to assess their ability to model transition. The next Chapter will describe the numerical methodology involved in the computational work of RANS, LES and hybrid RANS/LES modelling described in Chapter 5, Chapter 6 and Chapter 7 respectively.

Chapter 4

Numerical methodology

This Chapter describes the numerical framework employed to solve the governing equations of fluid flow together with the model transport equations. Most of the numerical aspects have been thoroughly described in a number of popular publications [94, 110, 111]. Therefore, for succinctness, only a brief overview will be provided in this Chapter, focusing specifically on the numerical aspects that are important and relevant to this work. The fluid flow solver employed in this work is OpenFOAM, which has been extensively used and thoroughly validated in a number of preceding studies [7, 101]. The finite volume method is employed to discretise the solution using the relevant discretisation schemes.

4.1 OpenFOAM

OpenFOAM (Open Field Operation and Manipulation) is an open source multi-physics modelling platform, written in C++ programming language that can perform numerical simulations of fluid flow, combustion, electro-magnetics, heat transfer, etc. OpenFOAM is very well known for its fluid flow modelling capabilities using the finite volume method (FVM) and it is currently widely used for CFD simulations. Unlike commercial CFD packages, OpenFOAM provides users access to the source code as well as the flexibility to modify the code, which enables researchers to develop, implement and test various new turbulence models, boundary conditions, etc. Due to this exclusive features, OpenFOAM is considered as a very powerful tool for researchers in academia and since its official release in 2004, it has been adopted by some engineering companies such as TATA Steel and Volkswagen AG[1]. All the CFD simulations and numerical implementations involved in this current work have been performed in OpenFOAM (version 2.3.x). A more detailed source of information about OpenFOAM can be accessed at www.openfoam.org.

4.2 The Finite Volume Method (FVM)

The finite volume method (FVM) is a discretisation method [111] for partial differential equations and in CFD, it is considered as one of the most adaptable discretisation method. Although the availability of other discretisation methods, FVM has the broadest applicability in both commercial and open-source CFD packages. The term ‘finite volume’ refers to the small volume around each

[1] https://www.esi-group.com/company/press/news-releases/second-openfoam-user-conference-be-held-berlin-oct-7-9-2014

nodal point in a computational grid. The FVM generally involves a series of methodological steps during the discretisation process. The first step is the decomposition of the domain into a finite number of subdomains (control volumes), and correlated nodes where the unknown variables are to be computed. Following domain decomposition, for each control volume (CV), the integral balance equations are formulated and approximated by numerical integration. Then, the derivatives and function values are approximated by interpolation with nodal values. In the FVM, volume integrals in a partial differential equation with a divergence term are transformed to surface integrals by applying the divergence theorem and are then evaluated as fluxes at each control volume. These methods are known to be conservative as the flux entering a control volume is equal to the flux exiting the adjacent one. A key advantage of the FVM is that it is applicable to complex geometries cases. A more in-depth description of the finite volume method, together with its mathematical formulation, applications and limitations can be found in [111].

4.3 Pressure-velocity coupling

The principle of FVM fluid flow solvers involves solving a set of coupled partial differential equations. For a three-dimensional case, u, v, w and p have to be computed together with any other additional parameters, for instance, temperature. When analysing the Navier-Stokes equations for a three-dimensional case, it is observed that there is a pressure term in each of the three momentum equations. These equations are discretised and can be solved for u, v, and w. A transport equation for pressure does exist as such and the continuity equation cannot be directly considered to achieve solution for p. Hence, pressure-velocity coupling algorithms that are referred to as iterative procedures, can be employed to overcome this problem by adjusting the pressure field, ensuring that the continuity equation is satisfied by the resulting velocity field. There are several available pressure-velocity coupling algorithm. The ones considered for the computations involved within this work, i.e. SIMPLE, SIMPLEC and PISO, are described as follows.

4.3.1 The SIMPLE algorithm

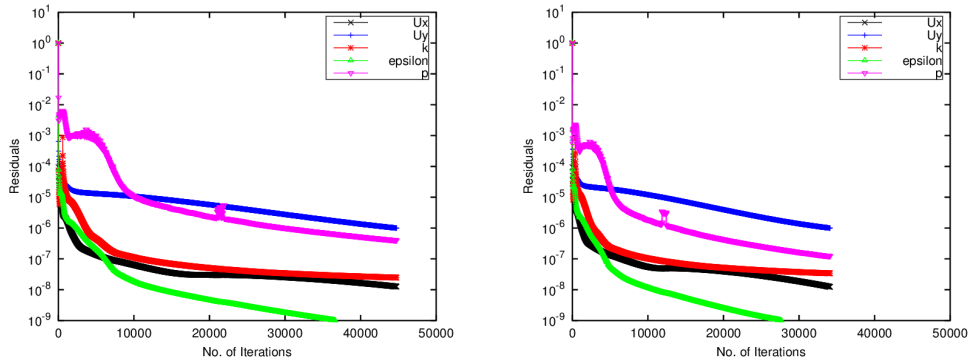
The SIMPLE (Semi-Implicit Method for Pressure Linked Equations) algorithm of Patankar [110] has been widely used in solving CFD problems. The SIMPLE algorithm functions based on the link between velocity and pressure corrections to impose the conservation of mass in order to obtain solutions for the pressure field, p. The SIMPLE algorithm operates by coupling the Navier-Stokes equations iteratively. The SIMPLE algorithm initially sets the boundary conditions. Then, the discretised momentum equation is solved to compute the intermediate velocity field. The mass fluxes at the cells faces are computed. The pressure equation is then solved and applied under relaxation followed by correction of mass fluxes at the cell faces. After the mass flux correction stage, the velocities are corrected based upon the new pressure field and the "final" step involves updating the boundary conditions. These steps are repeated as an iterative process until convergence is reached [112].

4.3.2 The SIMPLEC algorithm

The SIMPLEC (Semi-Implicit Method for Pressure Linked Equations-Consistent) algorithm of Van Doormal and Raithby [113] is an enhanced version of the SIMPLE algorithm, with a modification of the face flux correction expression. The SIMPLEC algorithm operates by the same iterative procedures that the SIMPLE algorithm employs. As a result of the modified correction expression, convergence is achieved quicker than in cases where pressure-velocity coupling is considered as the main deterrent to obtaining a converged solution. When using the SIMPLEC algorithm, the

relaxation factor for the pressure correction is assigned a magnitude of 1.0, which has proved to speed up convergence. However, a magnitude of 1.0 for the pressure correction parameter can sometimes make the solution unstable due to skewed cells in the computational grid.

Figure 4.1 illustrates the residual plots for the solutions of flow over a zero-pressure-gradient flat plate using the Launder-Sharma $k-\epsilon$ model and tested with the SIMPLE and SIMPLEC algorithms. In OpenFOAM, the simpleFoam steady-state solver has been implemented based upon the SIMPLE algorithm. In order to test the SIMPLEC algorithm, an in-house implementation was carried out as a steady-state solver in OpenFOAM called simplecFoam. The source code of the implemented SIMPLEC algorithm in OpenFOAM is available in Appendix B.



(a) Residuals Plot (SIMPLE) **(b)** Residuals Plot (SIMPLEC)

Figure 4.1: zero-pressure-gradient flat plate case tested with the Launder-Sharma $k-\epsilon$ model using the SIMPLE and SIMPLEC algorithms

The results indicated that the SIMPLEC solutions converged much faster (approximately 10,000 iterations less) than the SIMPLE solutions. With the SIMPLEC algorithm, the relaxation factor for the pressure correction was changed to 1.0, 0.7 for the velocity field and 0.4 for all the other fields. All the simulations converged until the final values of the dimensionless residual for all equations reached 10^{-6} (the set value) or lower.

4.3.3 The PISO algorithm

The PISO (Pressure-Implicit with Splitting of Operators) algorithm of Issa [114] in 1986. PISO is basically an extension of SIMPLE with an additional corrector step, involving an additional pressure correction equation to improve convergence [115]. Unlike the SIMPLE algorithm, the PISO algorithm does not solve all the equations in a sequence of iterations. Instead it operates by splitting the operators into an implicit predictor and corrector steps.The PISO algorithm performs the following steps: the predictor step, the 1st corrector step and the 2nd corrector step. The predictor step involves solving the momentum equation using the pressure field at t^{n-1} which yields an intermediate velocity field, which generally does not satisfy the continuity equation. The 1st corrector step involves solving the pressure equation and an intermediate pressure field is obtained, which is further applied to solve the momentum equation, yielding a velocity field, satisfying the continuity equation. A final velocity and pressure field is generated via the 2nd corrector step, which is simply a repetition of the 1st corrector step [94]. In OpenFOAM, the PISO algorithm is implemented as a transient solver called pisoFoam.

4.4 Summary of test cases

The 2D zpg flat plate cases [3] used throughout this current work are described within this section. For the developed hybrid RANS/LES, the model was further tested using a 2D zpg cylinder case [116], which is not described within this section.

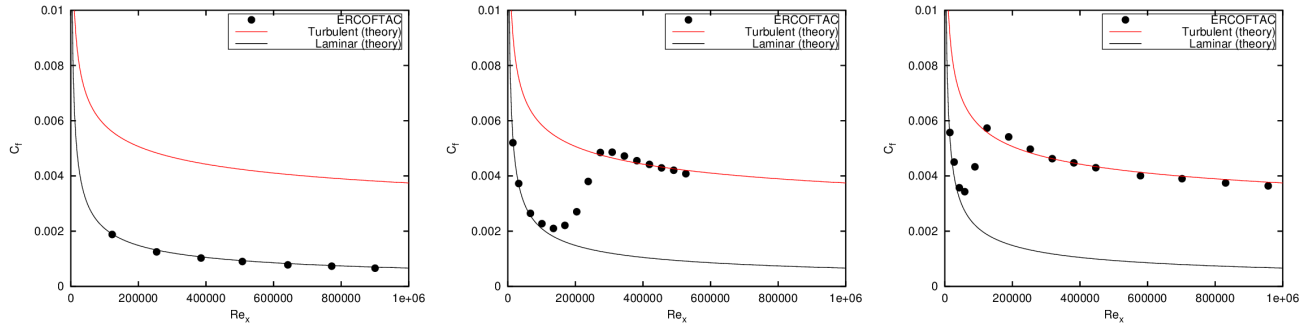
4.4.1 Experimental data: ERCOFTAC zero-pressure-gradient flat plate cases

The turbulence and transition models described in section 3.3 have been validated using the T3 series of experimental zero-pressure-gradient flat plate cases from the ERCOFTAC database. The three cases are represented by the abbreviated terms as T3A-, T3A and T3B. The freestream velocity, turbulence intensity and the kinematic viscosity varied for each of the cases. These are illustrated in table 4.1.

Table 4.1: Initial conditions of the experimental zero-pressure-gradient flat plate cases [3]

Test Case	$U_\infty (ms^{-1})$	Tu(%)	$\nu (m^2/s)$
T3A-	19.8	0.9	1.517×10^{-5}
T3A	5.4	3.0	1.513×10^{-5}
T3B	9.4	6.0	1.503×10^{-5}

The skin friction data of the T3A-, T3A and T3B cases plotted against the local Reynolds number, with reference to the laminar and turbulent theoretical profiles, Re_x are illustrated in figure 4.2.



(a) T3A-, Tu = 0.9%, Experimental **(b)** T3A-, Tu = 3.0%, Experimental **(c)** T3A-, Tu = 6.0%, Experimental

Figure 4.2: ERCOFTAC experimental results: skin friction coefficient C_f vs Re_x plots with reference to the theoretical laminar and turbulent profiles [3]

As discussed in Chapter 2, in order to alleviate the modelling accuracy limitation of RANS and the restrictive applications of LES as a result of the high computational cost, the hybrid RANS/LES method was proposed. A series of preliminary investigations was crucial in order to identify how different RANS models responds to transition. Additionally, it was indispensable to evaluate the performance of various SGS models, the influence of inlet boundary conditions on LES simulations and the grid requirements associated with LES method. These preliminary investigations, classified as crucial parts of the methodology towards achieving the aim of this work, are thoroughly described in Chapter 5 and 6.

Chapter 5

RANS: Assessment of selected RANS models in OpenFOAM

A range of high Reynolds number and low Reynolds number models (as described in Chapter 3) were tested on the 2D flat plate cases (as described in Chapter 4), in order to identify an ideal RANS turbulence model to be incorporated in the development of the proposed hybrid RANS/LES model. Whilst the RANS governing equations and the turbulence models have been described in Chapter 3, this section details the key pre-processing stages of the CFD simulations, the numerical setup and the numerical results comparison with the experimental data.

5.1 Computational grid

The computational grid is a discrete representation of the case geometry and it designates the cells on which flow will be solved numerically. Generating the computational grids is considered as one of the principal tasks of the pre-processing stage since the accuracy of the solution, the computational time and the convergence rate are heavily influenced by the grid resolution. Therefore, it is very important to achieve only the grid density required for a particular flow problem, taking into account the balance between accuracy and cost. To achieve this, a grid dependency study is usually carried out, which is a standard recommended procedure in CFD.

A diagram representing the different boundary names for the zero-pressure-gradient flat plate cases (T3A-, T3A and T3B) is illustrated below as figure 5.1. This diagram will be used as a reference to describe the grid dependency study as well as the numerical setup of the simulations.

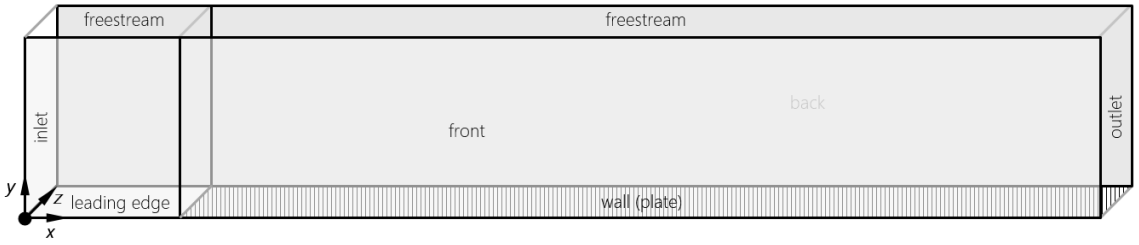


Figure 5.1: Sketch of the case geometry (2D flat plate)

5.1.1 Grid dependency study

Fürst [41] previously carried out a flat plate study and the ERCOFTAC experimental data [3] data was utilised for validation. The grid used by Fürst [41] produced satisfactory results, and hence it was considered as a baseline to create the initial low Reynolds number grid.

Although enough details of the computational grid were provided in [41] to accurately recreate it, a grid dependancy study was still carried out to ensure that higher accuracy could not be acheived through increasing the cell density in the x or y directions. The number of cells normal to the surface of the wall were tested by comparing the boundary layer velocity profiles (as illustrated in figure 5.2) produced by a slightly higher and slightly lower density of cells, and based on the results obtained, it was proven to be converging around 105 cells. The number of cells in the streamwise direction were evaluated by comparing skin-friction coefficient (C_f) plots as the number of cells was increased. This test in the streamwise direction was primarily carried out to evaluate the effect of aspect ratio of the grids on the simulations results. The grid as described by Fürst [41] produced a maximum aspect ratio of over 2.3×10^3, and therefore, the number of cells was increased to lower this value. Two additional cases were run for the test, one with a maximum aspect ratio of just under 1.0×10^3 and another with a maximum aspect ratio of just under 5×10^2. The three tested cases produced nearly identical results with negligable differences, due to which, the initial grid was not changed further.

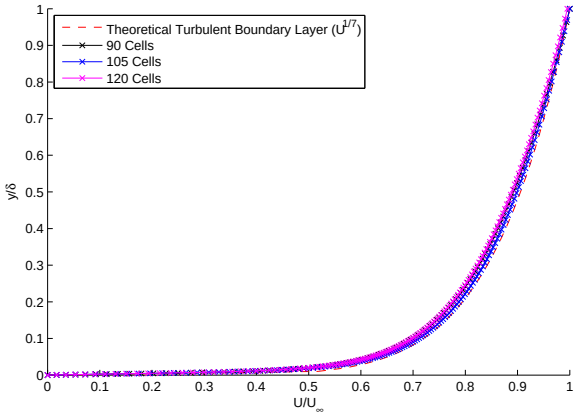


Figure 5.2: Boundary layer velocity profiles for varying grid densities, sampled $1.5m$ from the leading edge of the plate, Launder-Sharma $k-\epsilon$, 'T3A-' case

As illustrated in table 5.1, the same computational grid was used across all three cases for the low Reynolds number model studies and an individual grid was used for each case for the high Reynolds number models to allow the y^+ values to fall within the required range ($30 \leqslant y^+ \leqslant 100$). The only attribute of these grids that changed between the cases, were the wall-normal distance and the number of cells normal to the wall.

The wall-normal expansion for the high Reynolds number cases was required for the larger initial cell height and to allow the cells to expand with a reasonable cell-to-cell expansion ratio. The grids were created using the 'blockMesh' utility in OpenFOAM and were all comprised of two blocks, Block A dealt with all flow upstream of the flat plate and Block B contained the flat plate wall patch.

Table 5.1: Setup of meshes split into Block 1 (B1) and Block 2 (B2)

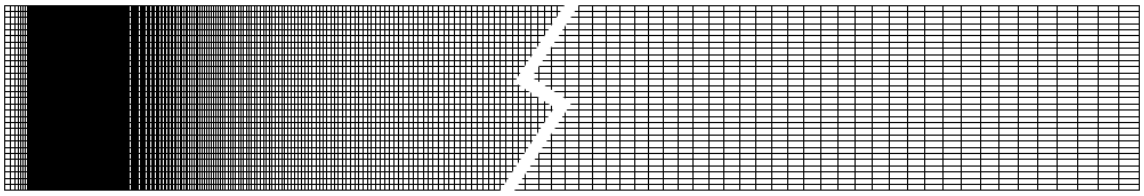
Case	Grid Resolution Block A	Dimensions Block A (m)	Grid Resolution Block B	Dimensions Block B (m)
Low Re case				
T3A-	35×105	0.05×0.175	600×105	2.9×0.175
T3A	35×105	0.05×0.175	600×105	2.9×0.175
T3B	35×105	0.05×0.175	600×105	2.9×0.175
High Re case				
T3A-	35×50	0.05×0.35	600×50	2.9×0.35
T3A	35×35	0.05×0.35	600×35	2.9×0.35
T3B	35×50	0.05×0.35	600×50	2.9×0.35

Table 5.2 shows that the maximum, minimum and average y^+ values for the high Reynolds number test case, T3A-, varied significantly when compared to the other two cases (T3A and T3B). The lower turbulence intensity (0.9%) is probably resulting in the large y^+ deviation over the surface of the plate, which is further supported by the fact that the T3B case has the least y^+ deviation along with the highest turbulence intensity of 6.0%. The total expansion ratio of each high Re grid was altered where possible to bring the y^+ ranges to more reasonable levels.

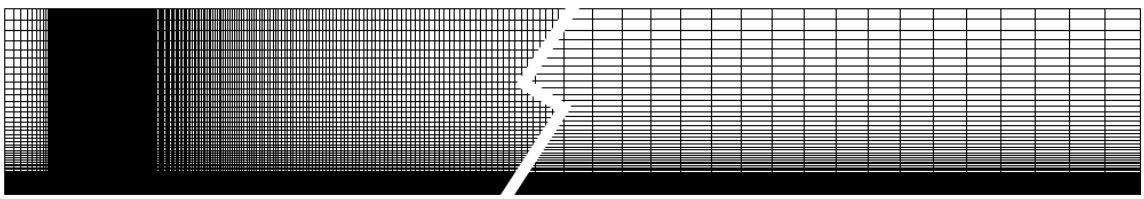
Table 5.2: The total expansion ratio ('simpleGrading') and the corresponding y^+ values

Case	simpleGrading (y-Direction)	Maximum y^+	Minimum y^+	Average y^+	y^+ Range
Low Re case					
T3A-	660.000	2.917	0.232	0.546	$y^+ \leqslant 1$
T3A	660.000	2.917	0.232	0.546	$y^+ \leqslant 1$
T3B	660.000	2.917	0.232	0.546	$y^+ \leqslant 1$
High Re case					
T3A-	17.448	40.780	4.754	23.598	$35 \leqslant y^+ \leqslant 100$
T3A	1.000	85.890	35.999	53.504	$35 \leqslant y^+ \leqslant 100$
T3B	4.794	48.973	36.305	41.607	$35 \leqslant y^+ \leqslant 100$

Figure 5.3 illustrates the computational grids for the RANS simulations which were generated in OpenFOAM using the 'blockMesh' utility. *Note: the computational grids illustrated are not to scale.*



(a) High Reynolds number computational grid



(b) High Reynolds number computational grid

Figure 5.3: Computational grids for RANS simulations

5.2 Numerical Setup

All the numerical simulations were performed in OpenFOAM 2.3.x. All the RANS turbulence models were available in OpenFOAM with the exception of the modified version of the $k_T - k_L - \omega$ model, for which an in-house implementation was required in OpenFOAM, by simply modifying the original version of the $k_T - k_L - \omega$ model available in OpenFOAM 2.3.x to incorporate the required changes [7]. The Reynolds-averaged Navier-Stokes equations along with the transport equations of the relevant turbulence model were initially solved using the incompressible, steady, turbulent SIMPLE solver, readily implemented and available in the OpenFOAM. The cases were then solved by employing the SIMPLEC algorithm, which was also an in-house implementation. A steady state time discretisation schemes was used and A central dicretisation scheme, *Gauss Linear* was used as the default settings of the gradient scheme. The divergence terms and the turbulent variables were computed using a second order bounded central discretisation scheme, *bounded Gauss linearUpwind*. The pressure equation was solved using the precondition conjugate gradient (PCG) solver, with a tolerance of 10^{12} and a relative tolerance of 0.01 between iterations. The remaining equations were solved using the precondition bi-conjugate gradient (PBiCG) solver together with the diagonal incomplete LU (DILU) asymmetric preconditioner, with a tolerance of 10^{10} and a relative tolerance of 0.1.

5.2.1 Boundary conditions

When solving the fluid flow governing equations, i.e. the Navier-Stokes equations, it is crucial to appropriately apply initial conditions and boundary conditions. While initial conditions defines the initial state of the fluid flow problem, the main function of boundary conditions is to provide information of the behaviour of the fluid at the boundaries of the computational domain. Both boundary conditions and initial conditions are very important parameters required in numerical fluid flow computations.

Subsequent to the grid dependency study and the selection of appropriate meshes for both the high and low Reynolds number cases, the boundary conditions were defined. Boundary conditions have strong influential effects on CFD simulations and they differ from model to model. Therefore, setting up the cases correctly, with the right boundary conditions for each turbulence model is very crucial.

Boundary conditions can be mathematically classified as follows:

1. Dirichlet Boundary conditions
2. Neumann Boundary conditions

The Dirichlet boundary condition involves with prescribing the value of a variable at a particular boundary of the computational domain. When applying a Neumann boundary condition, a *zeroGradient* is usually prescribed normal to the boundary. They are termed as *fixed value* and *ZeroGradient* in OpenFOAM.

The general boundary conditions applied to the physical boundaries of the computational domain are illustrated in table 5.3.

Table 5.3: General boundary conditions prescribed at the physical boundaries

Physical Boundary	Boundary Condition
Inlet	The velocity is prescribed at the inlet as a Dirichlet boundary condition (*fixedValue*(u_x u_y u_z)) and a Neumann boundary condition (*zeroGradient*) is prescribed for the pressure.
Outlet	At the outlet, a Dirichlet boundary condition is prescribed for the pressure as (*fixedValue*). A Neumann boundary condition is prescribed for the velocity as (*zeroGradient*).
Wall	At the wall a no-slip condition applies. The velocity of the fluid is given a Dirichlet condition (*fixedValue*) since it is equal to that of the wall. The pressure is assigned a *zeroGradient* condition as the flux through the wall is 0.
Front and Back	OpenFOAM generates and reads geometries in 3D and since these flat plate cases were 2D, it has to be instructed to solve in 2D by specifying an *empty* boundary condition to all the physical boundaries to which there is no flow in the normal direction and do not require the solutions to be computed.
Freestream	A Neumann boundary condition (zeroGradient) is prescribed for both velocity and pressure.
Leading Edge	This boundary was treated as an inviscid wall in order to stabilise the flow, before it reaches the viscous wall, i.e. the plate, providing a uniform profile at the leading edge. Hence, a *symmetryPlane* boundary condition was prescribed.

The ERCOFTAC zero-pressure-gradient flat plate experimental data [3] represented in table 5.4 below were prescribed as the initial conditions for computations with both the high and low Reynolds number models.

Table 5.4: Initial conditions for flat plate simulations at zero-pressure-gradient [3]

Test Case	$U_\infty/(ms^{-1})$	$Tu/(\%)$	$\nu/(m^2/s)$
T3A-	19.8	0.9	1.517×10^{-5}
T3A	5.4	3.0	1.513×10^{-5}
T3B	9.4	6.0	1.503×10^{-5}

All the turbulence models that were tested on the flat plate case involved the turbulent kinetic energy, k and either ϵ or ω as the specific dissipation rate. Hence, inlet and internal field boundary conditions were calculated using the standard equations (5.1), (5.2), (5.3) below, unless otherwise specified.

$$k = \frac{3}{2}(U_\infty Tu)^2 \tag{5.1}$$

$$\epsilon = C_\mu{}^{\frac{3}{4}} \frac{k^{\frac{3}{2}}}{l} \tag{5.2}$$

$$\omega = C_\mu{}^{-\frac{1}{4}} \frac{\sqrt{k}}{l} \tag{5.3}$$

The actual boundary conditions for the flat plate cases are illustrated in tables 5.5 to 5.11, taking into account the requirements of the various turbulence models. The calculated values presented in each table are for the T3A case. The boundary conditions will be the same for the T3B and T3A- for a particular turbulence model, with the exception of re-calculating the values due to different inlet conditions as per table 5.3.

Table 5.5: Boundary conditions for flat plate computations, $k-\epsilon$ model

	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**	**Leading Edge**
$\boldsymbol{U}$	fixedValue (5.4 0 0)	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{k}$	fixedValue 0.00394	zeroGradient	wallFunction 0.00394	empty	empty	zeroGradient	symmetryPlane
ϵ	fixedValue 0.00428	zeroGradient	wallFunction 0.00428	empty	empty	zeroGradient	symmetryPlane
ν_t	calculated 0	calculated 0	wallFunction 0	empty	empty	calculated 0	symmetryPlane

Table 5.6: Boundary conditions for flat plate computations, $k-\omega$ model

	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**	**Leading Edge**
U	fixedValue (5.4 0 0)	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient	symmetryPlane
p	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient	symmetryPlane
k	fixedValue 0.00394	zeroGradient	wallFunction 0.00394	empty	empty	zeroGradient	symmetryPlane
ω	fixedValue 1.208	zeroGradient	wallFunction 1.208	empty	empty	zeroGradient	symmetryPlane
ν_t	calculated 0	calculated 0	wallFunction 0	empty	empty	calculated 0	symmetryPlane

Table 5.7: Boundary conditions for flat plate computations, $k-\omega$ SST model

	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**	**Leading Edge**
$\boldsymbol{U}$	fixedValue (5.4 0 0)	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{k}$	fixedValue 0.00394	zeroGradient	wallFunction 0.00394	empty	empty	zeroGradient	symmetryPlane
ω	fixedValue 1.208	zeroGradient	wallFunction 1.208	empty	empty	zeroGradient	symmetryPlane
ν_t	calculated 0	calculated 0	wallFunction 0	empty	empty	calculated 0	symmetryPlane

Table 5.8: Boundary conditions for flat plate computations, Launder-Sharma $k-\epsilon$ model

	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**	**Leading Edge**
$\boldsymbol{U}$	fixedValue (5.4 0 0)	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{k}$	fixedValue 0.00394	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
ϵ	fixedValue 0.00428	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
ν_t	calculated 0	calculated 0	wallFunction 0	empty	empty	calculated 0	symmetryPlane

Table 5.9: Boundary conditions for flat plate computations, $q-\zeta$ model

	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**	**Leading Edge**
$\boldsymbol{U}$	fixedValue (5.4 0 0)	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{k}$	fixedValue 0.00394	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
ϵ	fixedValue 0.00428	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
ζ	fixedValue 0.02156	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{q}$	fixedValue 0.19841	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
ν_t	calculated 0	calculated 0	wallFunction 0	empty	empty	calculated 0	symmetryPlane

Table 5.10: Boundary conditions for flat plate computations, $v^2 - f$ model

	Inlet	**Outlet**	**Wall**	**Front**	Back	**Freestream**	**Leading Edge**
$\boldsymbol{U}$	fixedValue (5.4 0 0)	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{k}$	fixedValue 0.00394	zeroGradient	wallFunction 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
ϵ	fixedValue 0.00428	zeroGradient	wallFunction 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{v^2}$	fixedValue 0.02624	zeroGradient	wallFunction 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{f}$	zeroGradient	zeroGradient	wallFunction 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
ν_t	calculated 0	calculated 0	wallFunction 0	empty	empty	calculated 0	symmetryPlane

Table 5.11: Boundary conditions for flat plate computations, $k_T - k_L - \omega$ model

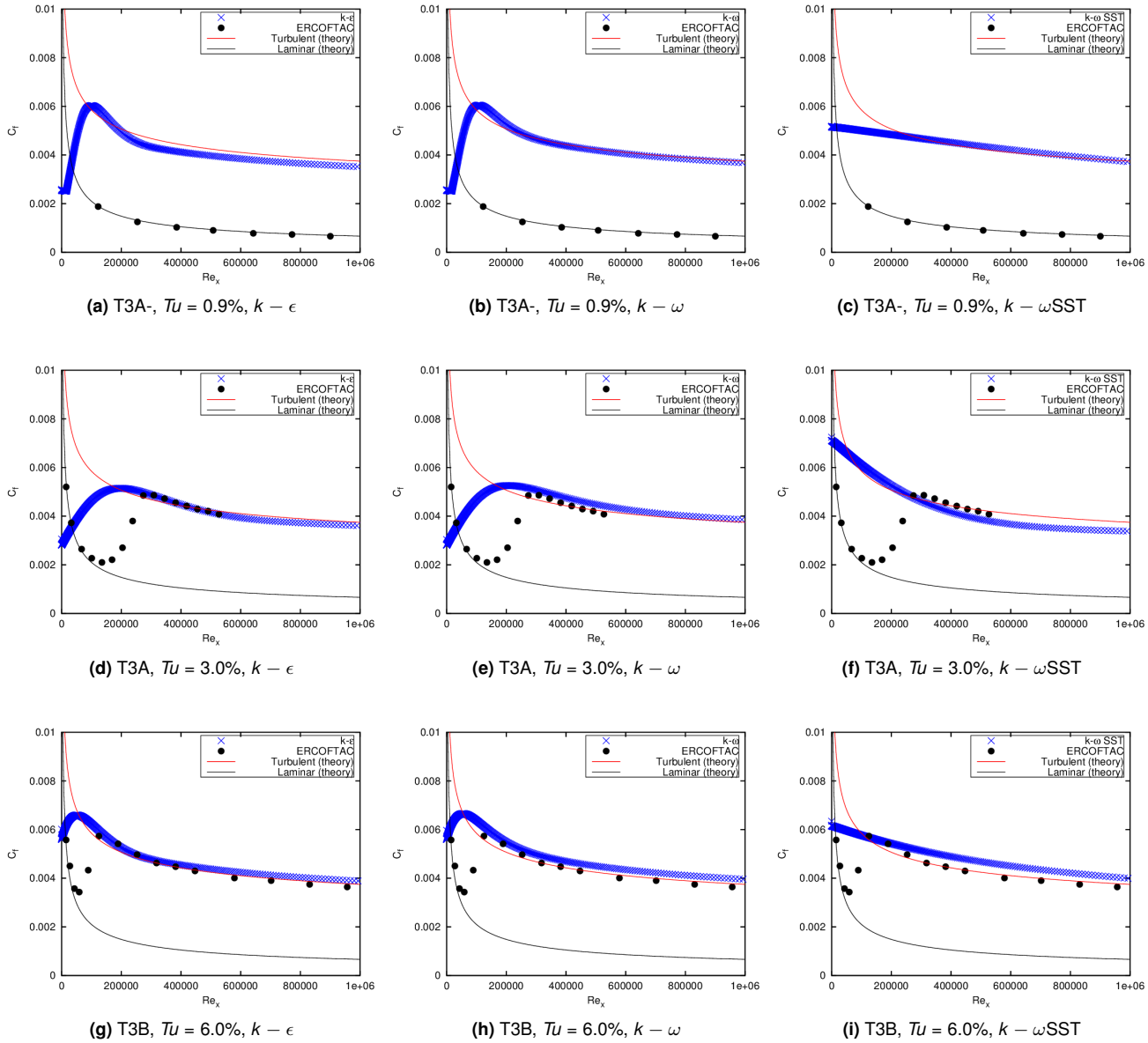
	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**	**Leading Edge**
$\boldsymbol{U}$	fixedValue (5.4 0 0)	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{k_T}$	fixedValue 0.00394	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{k_L}$	fixedValue 1×10^{-15}	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
ω	fixedValue 2.498	zeroGradient	zeroGradient	empty	empty	zeroGradient	symmetryPlane
ν_t	calculated 0	calculated 0	wallFunction 0	empty	empty	calculated 0	symmetryPlane

5.3 Results & discussion

Various CFD simulations were performed to assess the transition modelling capabilities of a series of high and low Reynolds number models. This section details the computational results, together with a thorough discussion on the performance of the various tested models, with a particular emphasis on their ability to predict laminar-turbulent transition and its location.

5.3.1 Results: High Reynolds number turbulence models

High Reynolds turbulence models were tested on the zero-pressure-gradient flat plate cases. The computational results are graphically illustrated below in figure 5.4, together with a comparison with the respective experimental and theoretical results. The blue lines represents the numerical results from CFD simulations, the black dots represents the ERCOFTAC experimental results [3] and the black and red solid lines represent the theoretical laminar and turbulent results respectively.



(a) T3A-, Tu = 0.9%, $k-\epsilon$ **(b)** T3A-, Tu = 0.9%, $k-\omega$ **(c)** T3A-, Tu = 0.9%, $k-\omega$SST

(d) T3A, Tu = 3.0%, $k-\epsilon$ **(e)** T3A, Tu = 3.0%, $k-\omega$ **(f)** T3A, Tu = 3.0%, $k-\omega$SST

(g) T3B, Tu = 6.0%, $k-\epsilon$ **(h)** T3B, Tu = 6.0%, $k-\omega$ **(i)** T3B, Tu = 6.0%, $k-\omega$SST

Figure 5.4: Comparison between numerical transition prediction of high Reynolds turbulence models at three turbulence intensity levels and the experimental results [3]

High Reynolds number turbulence models, when tested on the zero-pressure-gradient flat plate demonstrated different trends at varying turbulence intensities. At high turbulence intensity levels, i.e. T3A (Tu = 3.0%) and T3B (Tu = 6.0%), the $k-\epsilon$, $k-\omega$ and $k-\omega$ SST models demonstrated reasonably good agreement with the turbulent part of experimental data. However, at Tu = 0.9% (T3A-), the tested high Reynolds number models failed to model the laminar viscous sub-layer. Typical high Reynolds number models, such as the standard high Re $k-\epsilon$ and $k-\omega$ models, were developed based on fully developed turbulent flows.

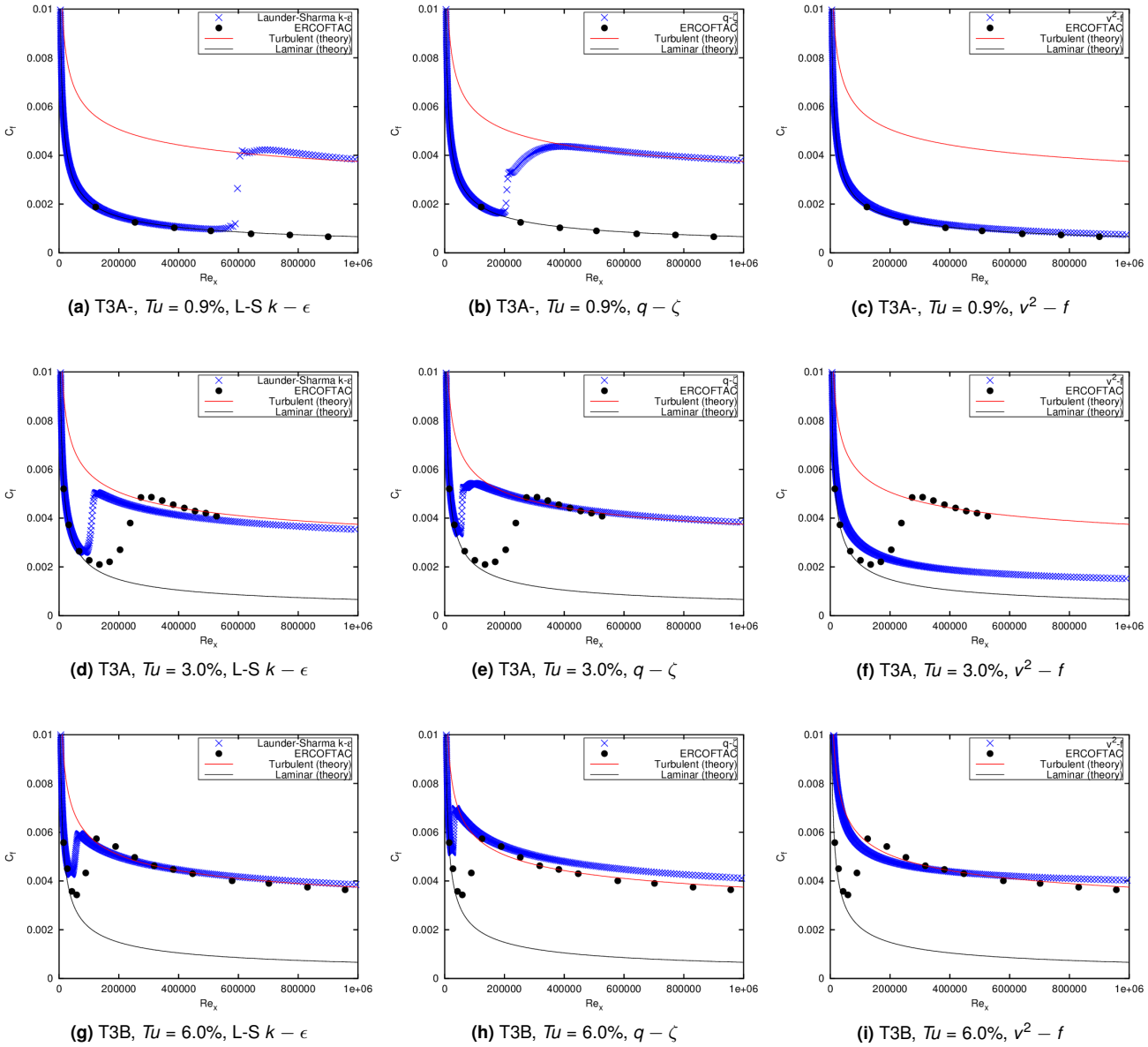
The $k-\epsilon$ high Reynolds number model uses a wall functions in order to preserve its near wall modelling capability with the viscous sublayer. The wall functions operate on the basis of the law of the wall rather than computing the solutions of the transport equations at the cells nearest to the wall. Although wall functions have demonstrated its feasibility to a wide range of engineering applications [117], the law of the wall is not relevant for cases involving laminar-turbulent transition and boundary layer separation. The standard $k-\epsilon$ turbulence model fails in resolving the viscous regime properly as a result of the assumption of the constant C_μ. Eventhough the $k-\omega$ high Reynolds number model can be integrated to the near-wall region, it is limited in terms of to asymtotic uniformity [117]. The $k-\omega$ model is associated with near wall predictive limitations, for instance, it under-predicts the turbulent kinetic energy, k in the viscous sublayer. The $k-\omega$ model is also known for its sensitivity to the assigned values of turbulent length scale [67]. Although Wilcox's [118] discussed the length scale sentivity as a positive attribute towards transition modelling, it is only at low FST levels that the boundary layer is influenced by the length scale [119]. When comparing the $k-\epsilon$ with the $k-\omega$ model, the latter is considered to perform better, but in terms of laminar-turbulent transition prediction, it suffers with significantly low prediction [118]. The $k-\omega$ SST model is well known for its developmental purpose and consists of a combination of the positive features of the $k-\epsilon$ and the $k-\omega$ models. Although it is well known to perform well with flows involving adverse pressure gradients and separation, it under-performs when applied to flows involving by-pass transition, by almost instantly triggering transition [120].

The numerical simulations results confirmed that high Reynolds number models have significant limitations to accurately predict laminar-turbulent transition. Hence, due to their poor performance, these models were not considered to be applicable to the development of the hybrid RANS/LES model and were disregarded.

5.3.2 Results: Low Reynolds number turbulence models

A range of low Reynolds number turbulence models were also tested on the zero-pressure-gradient flat plate cases. Low Reynolds number models were developed towards the attempt to overcome the limitation of high Reynolds number models of integrating to the wall. These models have the capability to resolve the flow in the viscous sublayer. The computational results are graphically illustrated in figure 5.5, together with a comparison with the respective experimental and theoretical results. The blue lines represents the numerical results from CFD simulations, the black dots represents the ERCOFTAC experimental results [3] and the black and red solid lines represent the theoretical laminar and turbulent results respectively.

The CFD simulations results indicated that unlike high Reynolds number models, low Reynolds models have the capability to predict transition to some degree. The Launder-Sharma $k-\epsilon$ and the $q-\zeta$ turbulence models indicated reasonable enough agreement with the the experimental results. However, the low Reynolds models tested failed to accurately predict the transition location.



(a) T3A-, Tu = 0.9%, L-S $k-\epsilon$ **(b)** T3A-, Tu = 0.9%, $q-\zeta$ **(c)** T3A-, Tu = 0.9%, v^2-f

(d) T3A, Tu = 3.0%, L-S $k-\epsilon$ **(e)** T3A, Tu = 3.0%, $q-\zeta$ **(f)** T3A, Tu = 3.0%, v^2-f

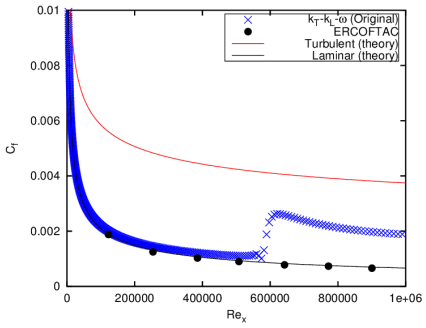
(g) T3B, Tu = 6.0%, L-S $k-\epsilon$ **(h)** T3B, Tu = 6.0%, $q-\zeta$ **(i)** T3B, Tu = 6.0%, v^2-f

Figure 5.5: Comparison between numerical transition prediction of low Reynolds turbulence models at three turbulence intensity levels and the experimental results [3]

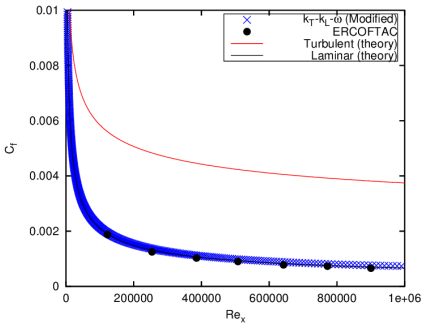
Transition location prediction being a crucial parameter in laminar-turbulent transition, the low Reynolds models are also disregarded towards the development of the DES model for transitional flows.

5.3.3 Results: transition model

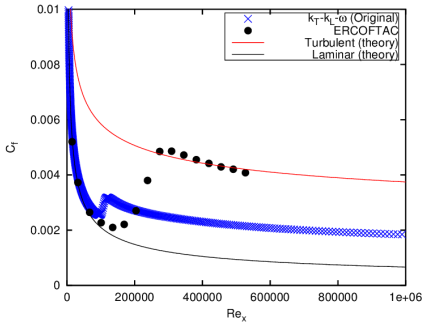
The original transition model of Walters and Cokljat [2], $k_T - k_L - \omega$ and the modified version of the same model were tested on the flat plate cases. The computational results are graphically illustrated in figure 5.6, together with a comparison with the respective experimental and theoretical results. The blue lines represents the numerical results from CFD simulations, the black dots represents the ERCOFTAC experimental results [3] and the black and red solid lines represent the theoretical laminar and turbulent results respectively.



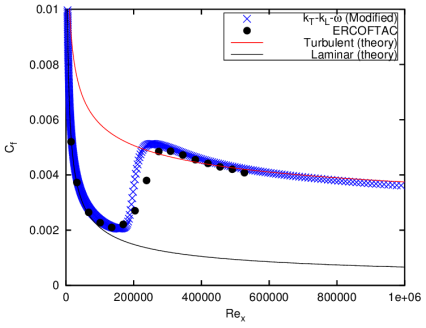
(a) T3A-, Tu = 0.9%, $k_T - k_L - \omega$ (original)



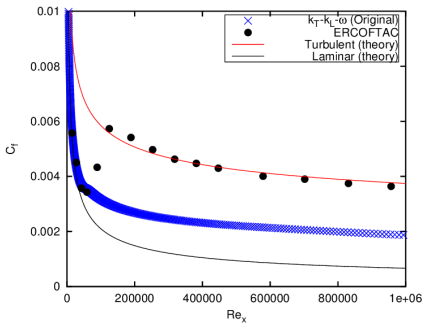
(b) T3A-, Tu = 0.9%, $k_T - k_L - \omega$ (modified)



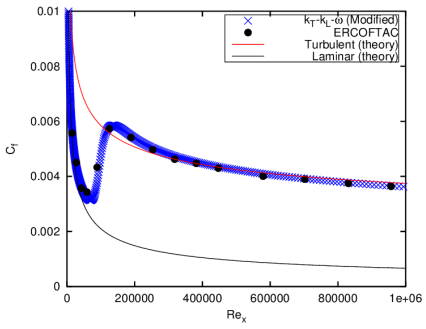
(c) T3A, Tu = 3.0%, $k_T - k_L - \omega$ (original)



(d) T3A, Tu = 3.0%, $k_T - k_L - \omega$ (modified)



(e) T3B, Tu = 6.0%, $k_T - k_L - \omega$ (original)



(f) T3B, Tu = 6.0%, $k_T - k_L - \omega$ (modified)

Figure 5.6: Comparison between numerical transition prediction of the original $k_T - k_L - \omega$ (as implemented in OpenFOAM) and the modified $k_T - k_L - \omega$ transitional RANS models at three turbulence intensity levels and the experimental results [3]

The original $k_T - k_L - \omega$ model of Walters and Cokljat [2] as implemented in OpenFOAM, highly under-predicted the transition from laminar to turbulence. However, the modified version of the model presented by Medina and Early [7] had very good agreement with the experimental data. The modified $k_T - k_L - \omega$ model captured the transition onset as well as the transition length. This model demonstrated good transition prediction capabilities and as a result, it will be considered as the background RANS model for the development of the proposed DES model.

Chapter 6

LES: Inflow conditions and grid sensitivity study

As part of the methodology for the development and implementation of the hybrid RANS/LES model, it was important to understand the principles of LES computations since the hybrid RANS/LES model will operate in LES mode in regions outside the boundary layer. This Chapter begins with a study comparing the response of the one-equation eddy viscosity model (*oneEqEddy*) and the dynamic one-equation eddy viscosity model (*dynOneEqEddy*), followed by an assessment of a range of inlet boundary conditions, which are readily available in OpenFOAM. Also, a grid resolution requirement investigation is carried out, where different computational grids are tested on a 2D zero-pressure-gradient flat plate to assess the sensitivity of LES solutions to the computational grid. Finally, a 3D LES simulation is presented, which was carried out, in order to assess the laminar-turbulent transition prediction capability of LES.

6.1 Numerical setup

The computational domain size for the 2D zero-pressure-gradient flat plate LES computations was $2.9m \times 0.175m \times 0.025m$ in the streamwise, wall-normal and spanwise direction respectively. Figure 6.1 is a sketch representation of the computational domain with the different boundary names. In OpenFOAM, the case geometry has to be specified in terms of 3D spatial coordinates and since the case is to be simulated in 2D, an "empty" boundary condition needs to be prescribed for the "front" and "back" boundaries (see figure 6.1). The computational grid was generated using the blockMesh utility in OpenFOAM, which is used to generate structured hexahedral grids. The freestream velocity, U_∞ and turbulence intensity, Tu prescribed at the inlet are $5.4m/s$ and 3.0% respectively.

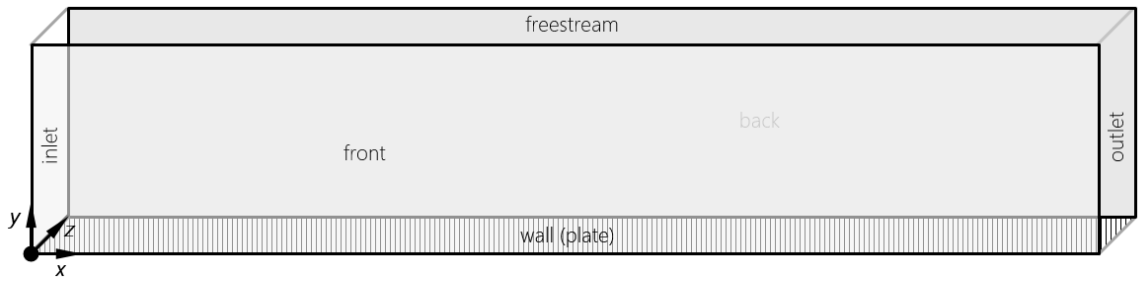


Figure 6.1: Sketch of the LES case geometry (2D flat plate)

All the SGS models, tested as part of this work, were readily available in OpenFOAM. The filtered Reynolds-averaged Navier-Stokes equations along with the transport equations of the relevant SGS models were solved using the incompressible, unsteady, turbulent pisoFoam solver, available in OpenFOAM. A second order implicit backward Euler (referred to as "backward" in OpenFOAM) time discretisation scheme was used and a central dicretisation scheme, *Gauss Linear*, was used to compute the gradient and the divergence terms. The divergence terms involving turbulent variables were computed using a limited central discretisation scheme, *Gauss limitedLinear 1*. The pressure equation was solved using the geometric algebraic multi grid *GAMG* solver, with *GaussSeidel* smoothing, a tolerance of 10^{-8} and a relative tolerance of 0 between iterations. The remaining equations were solved using the precondition bi-conjugate gradient (PBiCG) solver together with the diagonal incomplete LU *DILU* asymmetric preconditioner, with a tolerance of 10^{10} and a relative tolerance of 0. In order to achieve a stable solution, each time-step, Δt was automatically adjusted to maintain a Courant number, C_o below 1.

6.2 Boundary conditions

The general boundary conditions applied to the physical boundaries of the computational domain are illustrated in table 6.1. This table does not include the inlet boundary conditions since they will be described individually in section 6.4.

Table 6.1: General boundary conditions prescribed at the physical boundaries

Physical Boundary	Boundary Condition
Inlet	The different inlet boundary conditions used for the LES simulations are individually discussed in section 6.4.
Outlet	At the outlet, a Dirichlet boundary condition is prescribed for the pressure as (*fixedValue*). The *pressureInletOutletVelocity* boundary condition was prescribed for the velocity, which is only valid if a fixed value for the pressure is specified, i.e. a Dirichlet boundary condition
Wall	At the wall a no-slip condition applies. The velocity of the fluid is given a Dirichlet condition (*fixedValue*) since it is equal to that of the wall. The pressure is assigned a *zeroGradient* condition as the flux through the wall is 0.
Front and Back	OpenFOAM generates and reads geometries in three dimensional spatial coordinates and since the flat plate cases were 2D, an *empty* boundary condition is prescribed to all the physical boundaries to which there is no flow in the normal direction and do not require the solutions to be computed.
Freestream	A Neumann boundary condition (zeroGradient) is prescribed for both velocity and pressure.

The inlet velocity boundary condition was initialised as a laminar Blasius boundary layer profile. This was achieved by mapping the data from a separate RANS simulation (precursor simulation) to the LES inlet. This process is illustrated in figure 6.2.

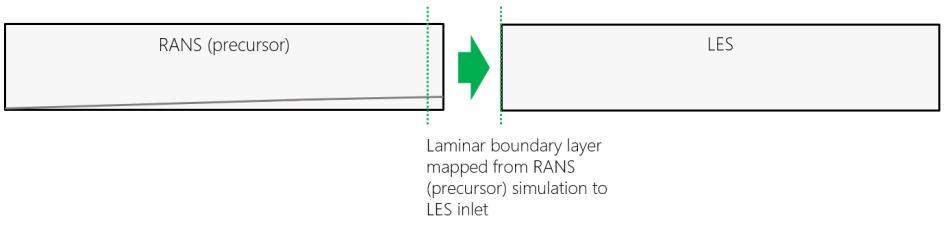
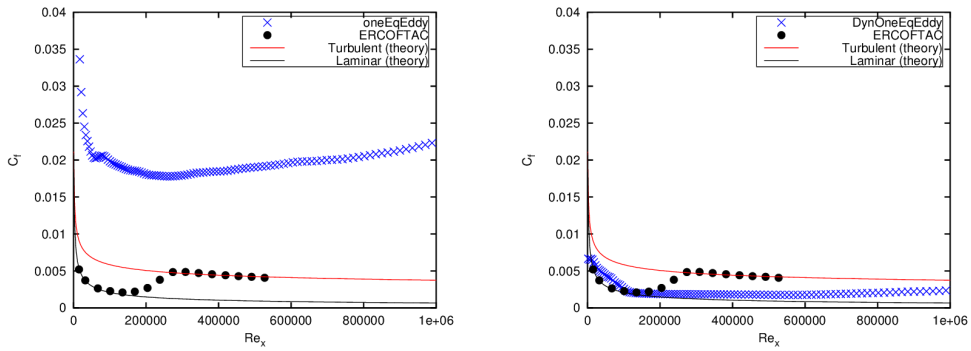


Figure 6.2: Mapping fields from RANS precursor simulation to LES simulation

6.3 Subgrid-scale (SGS) models test

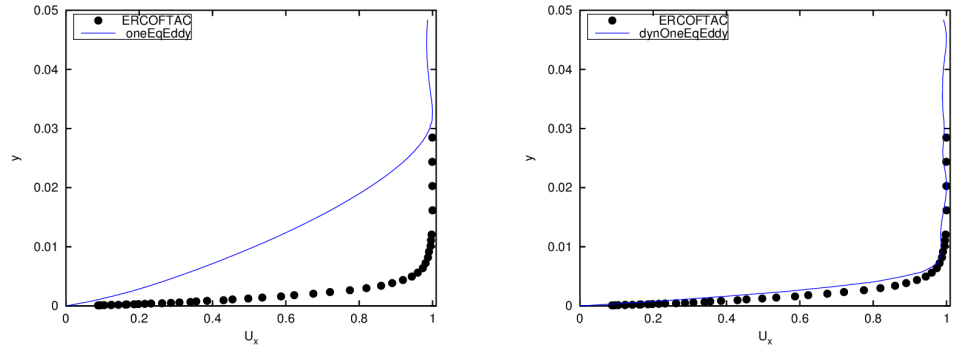
The *oneEqEddy* and *dynOneEqEddy* SGS models were tested on a 2D flat plate at zero-pressure-gradient. These tests were performed by prescribing the *turbulentInlet* boundary condition at the inlet (for the rest of the boundaries the boundary conditions were described as shown in table 6.1). The models were tested on a zero-pressure-gradient flat plate using a computational grid of $400 \times 340 \times 1$ cells in the streamwise, wall-normal and spanwise direction respectively. The numerical setup used is described in section 6.1. A comparison of the skin friction distribution along the streamwise direction of the plate, for each of the models tested, is illustrated in figure 6.3.



(a) one-equation eddy viscosity model **(b)** dynamic one-equation eddy viscosity model

Figure 6.3: SGS models comparison study - skin friction coefficient, C_f vs local Reynolds number, Re_x for a flat plate at zero-pressure-gradient (2D)

Figure 6.4 shows the laminar velocity profile (sampled at $0.5m$ from the inlet) for the two SGS models tested.



(a) one-equation eddy viscosity model **(b)** dynamic one-equation eddy viscosity model

Figure 6.4: SGS models comparison study - velocity profile, y vs U_x

Figures 6.3 and 6.4 show that neither of the SGS models tested captured laminar-turbulent transition. However, at this stage, the principal aim of these 2D simulations is to assess the performance of the *oneEqEddy* and *dynOneEqEddy* models. Therefore, for this particular study, the computational results will be compared to the theoretical laminar and turbulent skin friction coefficient solutions. The computational results demonstrated that using the same grid resolution, boundary conditions and simulation setup, the dynamic one-equation eddy viscosity model performs better than the standard one-equation eddy viscosity model in regards to their corresponding velocity profile predictions. This result is in agreement with finding presented by Sayadi and Moin [56], which involved the comparison between standard and dynamic SGS models (based on the Germano's dynamic procedure as described in section 3.3.5.4). Referring to figure 6.3,

the results indicate that the *oneEqEddy* SGS model over-estimated the skin friction by a factor of 8 in comparison to the *dynOneEqEddy* model. As a result, of its positive response, the dynamic model was selected for the remaining LES investigations.

6.4 Assessment of inflow conditions

The generation of effective and efficient inlet boundary conditions for LES is very challenging when simulating cases involving turbulent flows that involve inhomogeneities in the streamwise direction. Whilst a wide range of technical applications involved spatially developing boundary layers, it is necessary to have a developed turbulent field within the domain when simulating such cases using the LES approach. Therefore, the classical problem of inflow generation still remains an area of intense research. If conditions are not properly defined, it will usually result in inconveniently long domains, leading to an unnecessary waste of computational resources.

Computational solutions of LES are relatively sensitive to the inflow boundary conditions, particularly for cases such as plane jets [121], spatially developing boundary layers [122] and backward facing steps [123]. There are several approaches to achieve a developed field within the domain, two common methods are the synthesized turbulence method and the precursor simulation method. A detailed account on inlet boundary conditions for large-eddy simulations is presented in [124], where different inflow conditions are critically reviewed. In this work, the three different inflow conditions tested on a 2D zero-pressure-gradient flat plate are:

1. turbulent inlet
2. mapped inlet
3. oscillating inlet

The different inflow boundary conditions will be described and a brief discussion of the computational results will also be included (section 6.4.1, 6.4.2 and 6.4.3). All the three different inflow boundary conditions were tested using a computational grid of $630 \times 515 \times 1$ cells in the streamwise, wall-normal and spanwise direction respectively. The numerical setup as described in section 6.1 was used.

6.4.1 Turbulent inlet

The *turbulentInlet* boundary condition operates on the basis of generating a fluctuating inlet condition by incorporating a random component to a reference (mean) field [125]. It is classified as a synthetic inflow boundary condition, where random noise is introduced to the assigned velocity vector based on a defined turbulence level, also referred to as the fluctuation scale. This inlet boundary condition available in OpenFOAM and it has been implemented based on the mathematical formulation shown in equation (6.1).

$$x_p = (1 - \alpha)x_p^{n-1} + \alpha(x_{ref} + sC_{RMS}x_{ref}) \tag{6.1}$$

where x_p = patch values, x_{ref} = reference patch values, n = time level, α = fraction of new random component added to previous time value, C_{RMS} = RMS coefficient, and s = fluctuation scale.

The specific boundary conditions prescribed at all the boundaries are illustrated in table 6.2 for the 2D flat plate test case using the *turbulentInlet* boundary condition.

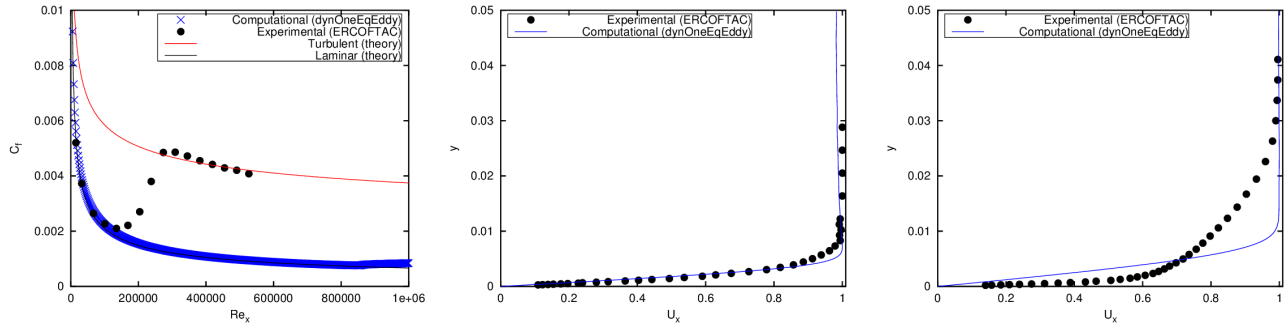
Table 6.2: Boundary conditions for the *turbulentInlet* inflow condition test on 2D zpg flat plate (T3A) using the *dynOneEqEddy* and *oneEqEddy* models

	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**
$\boldsymbol{U}$	turbulentInlet	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient
$\boldsymbol{k}$	fixedValue 0.0365	pressureInletOutletVelocity (0 0 0)	fixedValue 0	empty	empty	zeroGradient
ν_{sgs}	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient

The *turbulentInlet* inflow boundary condition for *U*, in OpenFOAM is specified as follows:

```
inlet
{
type turbulentInlet;
referenceField uniform (5.4 0 0);
fluctuationScale (0.02 0.01 0.01);
value uniform (5.4 0 0)
}
```

The results obtained from the simulations using the *turbulentInlet* inflow boundary condition are illustrated in figure 6.5.



(a) C_f vs Re_x **(b)** y vs Ux (laminar) **(c)** y vs Ux (turbulent)

Figure 6.5: Results from the *turbulentInlet* inflow boundary condition test on 2D zpg flat plate (T3A) using the *dynOneEqEddy* model

The computational results illustrated in figure 6.5 indicated that the *turbulentInlet* boundary condition failed to reproduce turbulent structures and the velocity fluctuations are damped out as it can be seen in figure 6.5. In an LES inflow boundary conditions investigation, de Villiers [101] tested the *turbulentInlet* boundary condition as well in channel flow simulations and the results demonstrated that the turbulent structures are not easily reproduced and that the amplitude of the fluctuations decreases very quickly, thereby dissipating turbulence from the computational domain.

6.4.2 Mapped inlet

The *mapped* inlet boundary condition is classified as a precursor type method of initialising the flow at the inlet. This inlet boundary condition involves assigning an offset distance, from which a plane is mapped back to the inlet (for all variables). Since the plane from which the quantities are being mapped is located within the same domain, it is advantageous in the sense that there is no need to use data from a pre-computed library, and therefore, it requires less storage space as a separate case is not used. Figure 6.6 illustrates a sketch of the mapping procedure involved when using the *mapped* inlet boundary condition.

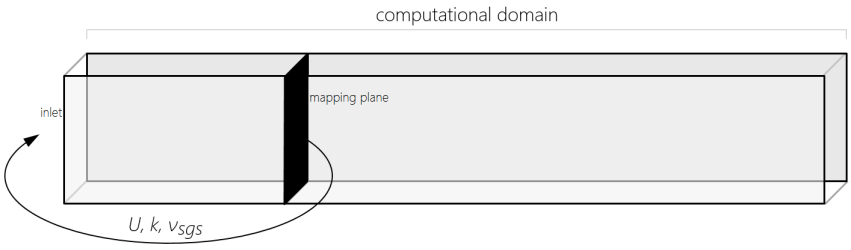


Figure 6.6: Internal mapping procedure of the *mapped* inlet condition

Table 6.3: Boundary conditions for the *mapped* inflow condition test on 2D zpg flat plate (T3A) using the *dynOneEqEddy* model

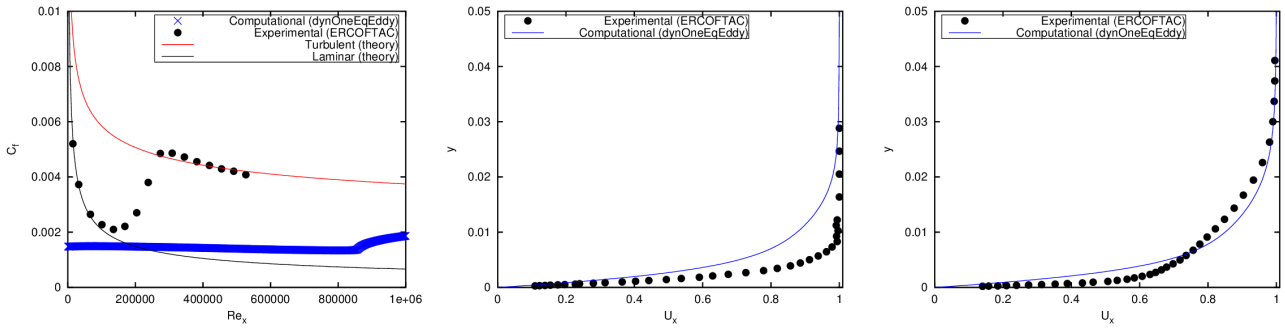
	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**
$\boldsymbol{U}$	mapped	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient
$\boldsymbol{k}$	fixedValue 0.0365	pressureInletOutletVelocity (0 0 0)	fixedValue 0	empty	empty	zeroGradient
ν_{sgs}	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient

The specific boundary conditions prescribed at all the boundaries are illustrated in table 6.3 for the 2D flat plate test case using the *mapped* inlet boundary condition.

The *mapped* inflow boundary condition for *U*, in OpenFOAM is specified as follows:

```
inlet
{
type  mapped;
value uniform (5.4 0 0);
interpolationScheme cell;
setAverage true;
average (5.4 0 0);
}
```

The results obtained from the simulations using the *mapped* inflow boundary condition are illustrated in figure 6.7.



(a) C_f vs Re_x **(b)** y vs Ux (laminar) **(c)** y vs Ux (turbulent)

Figure 6.7: Results from the *mapped* inflow boundary condition test on 2D zpg flat plate (T3A) using the *dynOneEqEddy* model

The results from figure 6.7 show that using the *mapped* inlet boundary condition, the solution throughout the computational domain is fully turbulent. This can be observed from the laminar and turbulent velocity profiles when sampling the results from both a laminar region (figure 6.7b) and a turbulent region (figure 6.7c) respectively. The likely reason of this unusual behaviour may be due to the fields being recycled from a turbulent station within the domain and being mapped to the inlet. Additionally, the skin friction distribution (figure 6.7a) throughout the domain demonstrate that this particular inflow condition will not be appropriate for the modelling requirements involved in this current work, i.e. laminar-turbulent transition.

6.4.3 Oscillating inlet

This boundary condition provides an oscillating condition in terms of amplitude and frequency as represented by the equation

$$x_p = [1 + asin(2\pi ft)]\, x_{ref} + x_o \tag{6.2}$$

Where x_p = patch values, x_{ref} = patch reference values, x_o = patch offset values, a = amplitude, f = frequency $(1/s)$, t = time (s).

The specific boundary conditions prescribed at all the boundaries are illustrated in table 6.4 for the 2D flat plate test case using the oscillating fixed value boundary condition.

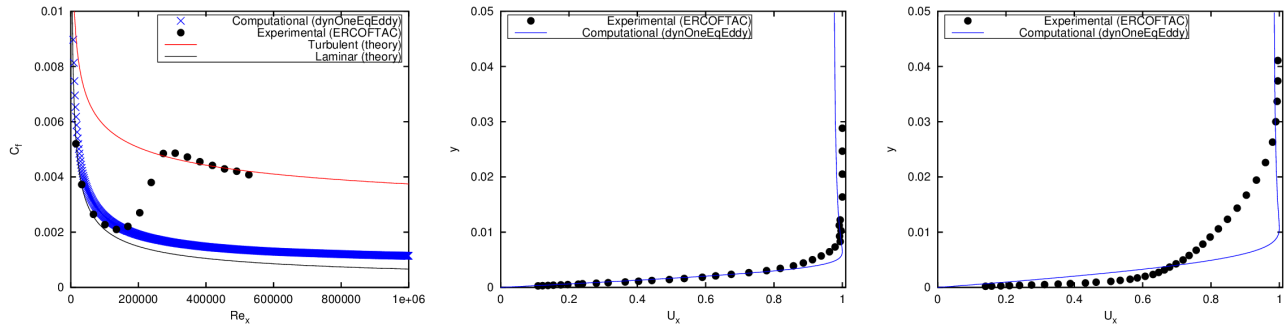
Table 6.4: Boundary conditions for the *oscillatingFixedValue* inflow condition test on 2D zpg flat plate (T3A) using the *dynOneEqEddy* model

	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**
$\boldsymbol{U}$	oscillatingFixedValue	zeroGradient	fixedValue (0 0 0)	empty	empty	zeroGradient
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient
$\boldsymbol{k}$	fixedValue 0.0365	pressureInletOutletVelocity (0 0 0)	fixedValue 0	empty	empty	zeroGradient
ν_{sgs}	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient	zeroGradient

The oscillating fixed value boundary condition at the inlet, in OpenFOAM is specified as follows:

```
inlet
{
type oscillatingFixedValue;
value uniform (5.4 0 0);
refValue uniform (5.4 0 0);
offset (0.35 0 0);
amplitude constant 0.35;
frequency constant 250;
}
```

The results obtained from the simulations using the *oscillatingFixedValue* inflow boundary condition are illustrated in figure 6.8.



(a) C_f vs Re_x **(b)** y vs Ux (laminar) **(c)** y vs Ux (turbulent)

Figure 6.8: Results from the *oscillatingFixedValue* inflow boundary condition test on 2D zpg flat plate (T3A) using the *dynOneEqEddy* model

The results obtained from the *oscillatingFixedValue* inflow condition test as illustrated in figure 6.8 shows a similar behaviour to the *turbulentInlet* boundary condition. Turbulence within the computational domain is damped out.

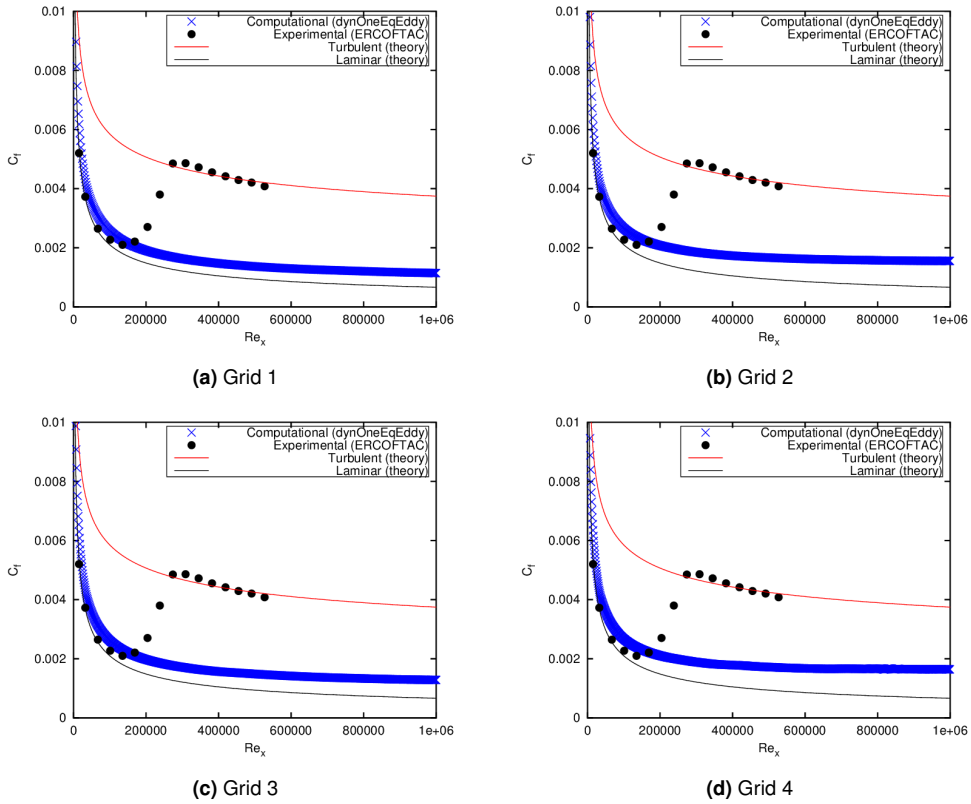
6.5 Grid resolution assessment

A grid resolution assessment for the LES computations have been performed in order to study the sensitivity of the LES solutions to the computational grid. Four computational grids with different resolutions (as described below in table 6.5) were tested on a 2D zero-pressure-gradient flat plate case using the dynamic one-equation eddy viscosity model. The inlet boundary condition specified for the LES computations was the *oscillatingFixedValue* and the remaining boundaries were assigned the boundary conditions as described in table 6.4. The simulation cases were prescribed the finite volume discretisation schemes and the finite volume solvers as described in section 6.1.

Table 6.5: Computational grids resolution details tested for the LES grid sensitivity study

	Grid 1	**Grid 2**	**Grid 3**	**Grid 4**
No. of cells (*x*)	630	740	900	1100
No. of cells (*y*)	515	530	550	570
y^+	0.26	0.23	0.19	0.15

Figure 6.9 illustrates the results obtained from the grid resolution study, in terms of the skin friction coefficient C_f plotted against the local Reynolds number, Re_x.



(a) Grid 1 **(b)** Grid 2

(c) Grid 3 **(d)** Grid 4

Figure 6.9: Results from the LES grid resolution study using the *oscillatingFixedValue* boundary condition and the *dynOneEqEddy* model

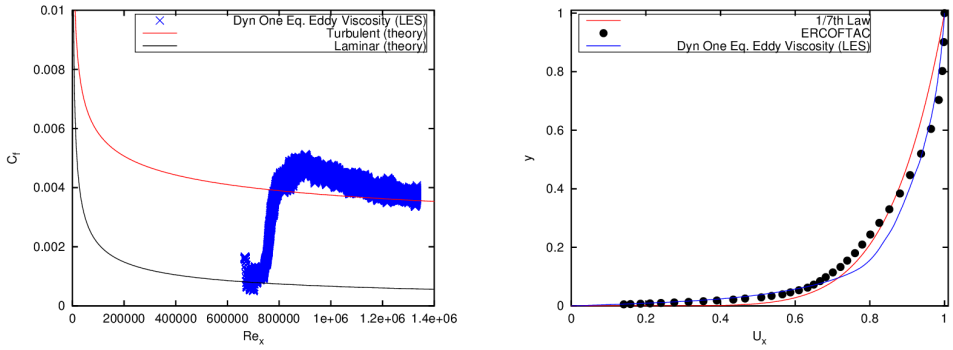
The results obtained from the grid resolution study indicate that the solution remains laminar throughout the computational domain, regardless of the grid resolution. However, referring to figure 6.9, it is observed that the grid with the lowest resolution, i.e. Grid 1, had the best agreement with the laminar solution when compared with the other grids. After further investigation by reviewing the LES source code in OpenFOAM, it was identified that as a result of the definition of the LES filter used, i.e. the cubeRootVol, $\Delta = (\Delta_x \Delta_y \Delta_z)^{1/3}$, stretching the grid in the wall-normal direction can have a significant influence on the computational solution. This needs to be further investigated.

6.6 3D flat plate LES

From all the LES studies performed as part of this current work, it can be concluded that LES fails to model laminar-turbulent transition in 2D. Therefore, a 3D simulation was performed to further assess the laminar-turbulent transition modelling capability of LES.

The computational domain size for the 3D zero-pressure-gradient flat plate LES computations was $1.0m \times 0.02m \times 0.01m$ in the streamwise, wall-normal and spanwise direction respectively. The simulation setup was based on the numerical settings described in section 6.1. The boundary conditions prescribed were the same as those described in table 6.4, with the exception of the front and back boundaries, which were prescribed as cyclic boundaries. The inflow condition prescribed at the inlet was the *oscillatingFixedValue* and the SGS model used was the *dynOneEqEddy*. The simulations were performed at a freestream velocity, U_∞ of $10m/s$.

The results obtained from the 3D LES computations are illustrated in figure 6.10.



(a) C_f vs Re_x **(b)** y vs Ux

Figure 6.10: Results from the 3D LES on a zero-pressure-gradient flat plate

The skin friction distribution over the flat plate is illustrated in figure 6.10a and it shows the transition of the laminar boundary layer to turbulent. Additionally, when comparing the velocity profile sampled at a turbulent station, the results show a very good agreement with the experimental results as well as the 1/7th Power Law.

Figure 6.11 shows iso surfaces based on the computed Q-criterion. The inflow structure resembles T-S waves and were generated by the oscillating boundary condition at a frequency of $250Hz$ to match typical natural transition T-S frequencies, the T-S waves breakdown to form gamma and hairpin vortices and then breakdown into fully turbulence flow.

From the LES studies, it can concluded that LES is difficult to set up (especially the inflow conditions) but it does provide a huge amount of information about the nature of the flow and its structures.

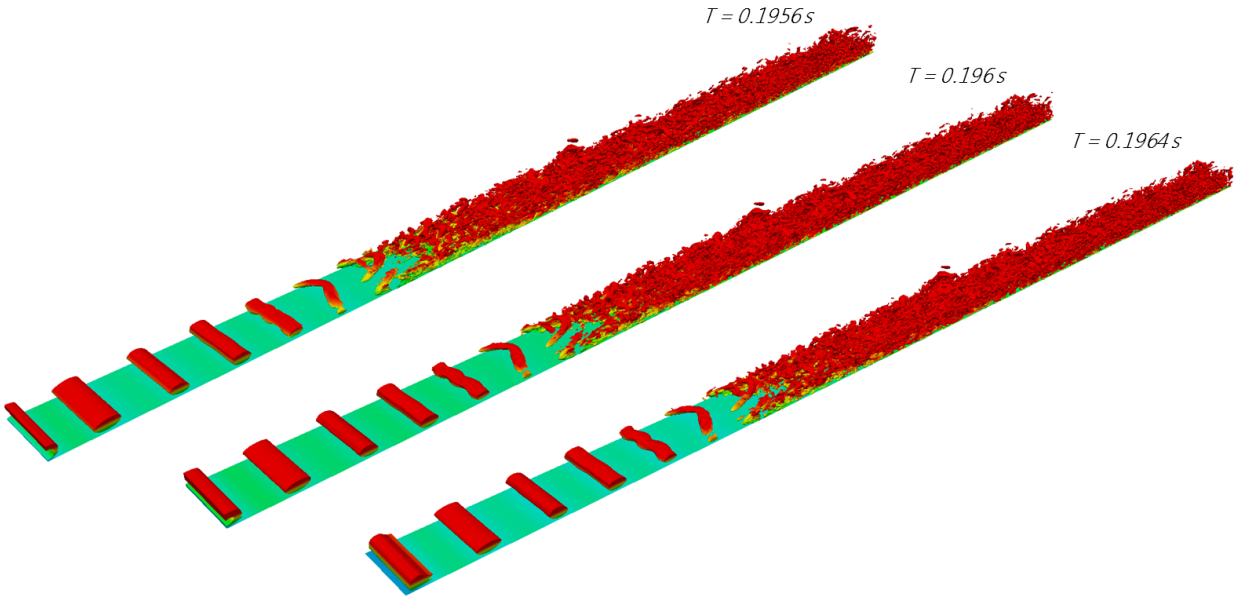


Figure 6.11: Boundary layer velocity profiles for varying grid densities, sampled $1.5m$ from the leading edge of the plate, Launder-Sharma $k-\epsilon$, 'T3A-' case

Chapter 7

DES: Proposed hybrid RANS/LES model

The development of a hybrid RANS/LES model for transitional flows is described in this Chapter. The hybridisation technique employed is the detached-eddy simulation (DES), which was selected on the basis of the investigation of different existing hybridisation techniques, as described in Chapter 2. In fact, the DES approach was introduced in 1997 [1] to address the challenge of modelling high Reynolds number and massively separated flows. The DES approach employs a RANS turbulence model as the background model, which is modified in such a way to introduce the capability of switching to a SGS model in the regions of massively separated flow and reverting back to the original RANS model in the attached boundary layer regions. Switching between RANS and LES mode is controlled/determined using the local grid cell size. From the RANS models study in Chapter 5, the $k_T - k_L - \omega$ transition model of Walters and Cokljat [2], which was modified by Medina and Early [7], demonstrated good response to laminar-turbulent transition when tested on a 2D zero-pressure-gradient flat plate. As a result, the $k_T - k_L - \omega$ model was selected as the background RANS model for the proposed hybrid RANS/LES model formulation.

This Chapter will provide a detailed account of the DES approach, beginning with a brief introduction, followed by a detailed description of the original DES model formulation [1]. A summary of previously developed hybrid RANS/LES models based on the DES methodology with different background RANS models will also be covered within this Chapter. This Chapter focuses mainly towards the mathematical formulation of the proposed DES model (hereinafter referred to as the $\boldsymbol{k_T - k_L - \omega}$ **DES** model). Since the standard approach showed deficiencies in correctly switching between RANS and LES modes, a final model is proposed based on the delayed detached-eddy simulation (DDES) (hereinafter referred to as the $\boldsymbol{k_T - k_L - \omega}$ **DDES** model). A detailed description of the model implementation in OpenFOAM, followed by the numerical methodology associated within the numerical testing phase of the models are also presented. Finally, the results obtained from the numerical tests of the new DES model will be presented, together with a thorough discussion.

7.1 Detached-eddy simulation: An introduction

Detached-eddy simulation (DES) is a variant of the hybrid RANS/LES modelling approaches that was proposed by Philippe R. Spalart [1]. This method, since its formal introduction in 1997, has continuously witnessed a widespread application both within academia and industry. Figure 7.1 illustrates the key factors contributing towards the motivation of the DES approach.

Figure 7.1: Motivational factors of the DES approach [104]

The DES approach involves treatment of the near-wall regions in RANS mode and the rest of the flow (that involves mainly detached regions) in LES mode, using a single turbulence model. The original DES model of Spalart et al. [1] is based on the one-equation Spalart-Allmaras (SA) RANS model [60]. This particular hybrid RANS/LES approach is considered to be more implementation friendly in contrast to other existing hybrid modelling approaches, since it uses the same underlying RANS model with different length scales (RANS or SGS) depending on the local grid size. An excellent definition of detached-eddy simulation provided by Travin et al.[126] is quoted before advancing to the core of this work:

> *"A detached-eddy simulation is a three-dimensional unsteady numerical solution using a single turbulence model, which functions as a subgrid-scale model in regions where the grid density is fine enough for a large-eddy simulation, and as a Reynolds-averaged model in regions where it is not."*

DES is one of the most widely used hybrid RANS/LES methods and has witnessed major successes [66, 126, 127]. While the scope of this current work is limited to the detached-eddy simulation approach, the interested reader is referred to the DESider[1] [68] and FLOMANIA[2] [128] reference sources, where a wide range of hybrid RANS/LES techniques are described.

7.2 The original DES model

The original DES model of Spalart et al. [1] is a combination of the Spalart-Allmaras (SA) one-equation RANS model [60] and its equivalent subgrid version using a DES limiter. Before describing the DES model of Spalart et al. [1], its background RANS model is briefly described.

A one-equation eddy viscosity RANS model was proposed by Spalart and Allmaras [60], which employs a transport equation to model the eddy viscosity. The transport equation reads

$$\frac{\partial \tilde{\nu}}{\partial t} + \tilde{u}_j \frac{\partial \tilde{\nu}}{\partial x_j} = \underbrace{c_{b_1} \widetilde{S} \tilde{\nu}}_{\text{Production}} + \underbrace{\frac{1}{\sigma}\left[\frac{\partial}{\partial x_j}\left((\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_j}\right) + c_{b_2}\frac{\partial \tilde{\nu}}{\partial x_j}\frac{\partial \tilde{\nu}}{\partial x_j}\right]}_{\text{Diffusion}} - \underbrace{c_{\omega_1} f_\omega \left(\frac{\tilde{\nu}}{d}\right)^2}_{\text{Destruction}} \tag{7.1}$$

The eddy viscosity, ν_t is computed from

$$\nu_t = \tilde{\nu} f_{v_1} \qquad f_{v_1} = \frac{\chi^3}{\chi^3 + C_{\nu 1}^3} \qquad \chi = \frac{\tilde{\nu}}{\nu} \tag{7.2}$$

$$\widetilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v_2} \qquad f_{v_2} = 1 - \frac{\chi}{1 + \chi f_{v_1}} \tag{7.3}$$

[1] DESider is a European Commission funded project, that aimed towards demonstrating the capabilities of hybrid RANS/LES approaches, particularly DES.

[2] FLOMANIA is a European Union funded project, that aimed towards the development of enhanced turbulence models for industrial applications.

$$f_w = g\left[\frac{1+C_{w_3}^6}{g^6+C_{w_3}^6}\right]^{1/6} \qquad g = r + C_{w_2}(r^6 - r) \qquad r = \frac{\tilde{\nu}}{\widetilde{S}\kappa^2 d^2} \tag{7.4}$$

A detailed description of the Spalart-Allmaras model can be found in [60], where all the functions and the model constants are documented.

The original DES model was formulated by substituting the distance function d from the destruction part of the SA transport equation (7.1) with a modified distance function, which is essentially the DES limiter and can be mathematically defined as

$$l_{DES} = \min\{d_w, C_{DES}\Delta\} \tag{7.5}$$

where l_{DES} is the model length scale, d_w is the wall distance involved in the destruction term of the SA model, C_{DES} is the model empirical constant and Δ is the maximum length of the local grid size.

$$\Delta = \max\{\Delta_x, \Delta_y, \Delta_z\} \tag{7.6}$$

The DES model of Spalart et al. [60] is derived when the wall distance term, d in the destruction term of the RANS transport equation (7.1), is simply substituted by the DES length scale defined by (7.5), performing as the baseline RANS model (i.e. SA model) in the attached regions (the near-wall regions) where $d_w < C_{DES}\Delta$ and as an SGS model in the detached regions (away from the wall) by applying the filter $C_{DES}\Delta$ where $d_w > C_{DES}\Delta$.

7.2.1 Generalisation of the DES limiter to other RANS models

The applicability of the DES approach to other RANS models (rather than the SA model) has proven to be feasible over the years. Travin et al. [129] proposed a generalised definition of the DES limiter, which is given by

$$l_{DES} = \min\{l_{RANS}, l_{LES}\} \tag{7.7}$$

where l_{RANS} is the RANS scale, which is defined for the $k-\epsilon$ and the $k-\omega$ based RANS models as $l_{RANS} = k^{1.5}/\epsilon$ and $l_{RANS} = \sqrt{k}/\left(C_\mu\omega\right)$ respectively. The subgrid length scale l_{LES} is defined as $l_{LES} = C_{DES}\Delta$.

It is to be noted that based on the generalised definition of l_{DES} in (7.7), the RANS/LES interface may become dependent on the computational solution, unlike being only grid-dependent when l_{DES} is defined as in (7.5). However, the definition of l_{DES} in (7.7) offers the flexibility in regards to choosing which term within the background RANS transport equation, the RANS length scale is to be substituted by the DES length scale.

Due to this generalised definition of the DES length scale, several improved DES models have been successfully developed using other RANS models and have been successfully applied to various cases. In fact, the very first DES implementations based on other background RANS models were first demonstrated by Strelets [66] and Travin et al. [129], which involved the DES formulation based on the $k-\omega$ SST model of Menter [67].

A summary of various DES formulations based on different background RANS models, which was published in [68] and [104] are illustrated in table 7.1, with the acronym by which they are usually recognised, followed by the transport variables and their respective references.

Table 7.1: DES models developed based on different RANS models [68, 104]

Background RANS model	Acronym	Transport variable(s)	References
Spalart-Allmaras model	SA DES *(original model)*	$\tilde{\nu}$	Spalart et al. [1]
Spalart-Allmaras model with Edwards modification	SAE DES	$\tilde{\nu}$	Edwards et al. [130]
Strain-adaptive linear Spalart-Allmaras model	SALSA DES	$\tilde{\nu}$	Rung et al. [131]
Spalart-Allmaras model	SA DES *(with low Reynolds number correction)*	$\tilde{\nu}$	Shur et al. [132] Spalart et al. [74]
Spalart-Allmaras model	SAE DES *(with low Reynolds number correction)*	$\tilde{\nu}$	Mockett et al. [133]
Spalart-Allmaras model	SALSA DES *(with low Reynolds number correction)*	$\tilde{\nu}$	Mockett et al. [133]
Spalart-Allmaras model	Zonal SA DES *(with low Reynolds number correction)*	$\tilde{\nu}$	Mockett et al. [133]
Menter $k-\omega$ SST model	M-SST DES	k, ω	Strelets [66] Travin et al. [129]
Wilcox $k-\omega$ model	WCX DES	k, ω	Yan et al. [134]
Linear local realisable $k-\omega$ model	LLR DES	k, ω	Rung et al. [131]
Turbulent/non-turbulent (TNT) $k-\omega$ model	X-LES	k, ω	Kok et al. [135]
Wilcox $k-\omega$ model	OEM DES	k, ω	Braza et al. [136] El Akoury et al. [137]
Modified Chen and Patel $k-$ *equation* model	k-DES	k	Peng [138]
Compact explicit algebraic stress model	CEASM DES	k, ϵ	Lübcke et al. [139]
Lien-Leschziner $k-\epsilon$ model	LL DES	k, ϵ	Lien et al. [140]
$\gamma - Re_\theta$ transition model	$\gamma - Re_\theta$ DES	γ, Re_θ	Sørensen et al. [69]

7.3 Limitations of the DES approach

Despite the successful development of a range of DES models based on different RANS models, which have demonstrated positive outcomes from the simple to complex aerodynamic applications, some serious limitations of the DES approach have been reported. These limitations are considered to be grid-dependent and are mainly: the activation of near-wall damping in the LES region of the computational domain, the incursion of LES mode in the boundary layer and the "grey area" problem.

In cases where the boundary layers are very thin with very fine grid, DES operates in the RANS mode and in separated regions where the grid is nearly isotropic, it exhibits the subgrid charac-

teristics, operating in LES mode. However, in cases involving thick boundary layers and weak separation, DES models tend to have an unusual behaviour, which is due to the computational grid spacing parallel to the wall being smaller than the boundary layer thickness, δ. In fact, the grid is so fine that the DES model applies the LES characteristics, according to the definition of the DES length scale in equation (7.7), thereby reducing the eddy viscosity well below than that of the RANS level, without replacing the modelled Reynolds stresses by those originated from the SGS velocity fluctuations. Therefore, the stresses are significantly lowered, which introduces the problem of grid-induced separation (GIS). This problem can be explained by considering the three different types of computational grid in the attached boundary layer region, as illustrated in figure 7.2. The top grid (type A) represents a natural DES grid, the bottom left grid (type B) represents a ambiguous grid spacing and the right grid (type C) represents a LES type grid. The dotted lines represents the mean velocity profile.

Figure 7.2: Computational grid within a boundary layer [74]

With reference to the type A grid, the grid spacing parallel to the wall set the maximum local grid spacing, Δ based on equation (7.6), which exceeds δ so that the DES model operates in RANS mode ($l_{DES} = d_w$) within the boundary layer. In such cases, this behaviour of the DES model is correct since the DES approach was developed to circumvent LES within the attached boundary layer region.

At the other extreme, the type C grid is very fine and is classified as an LES type grid, with the spacings smaller than δ. Due to the nature of the grid and taking into account the definition of the DES length scale, the DES model operates in LES mode ($l_{DES} = C_{DES}\Delta$) over the majority of the boundary layer and in RANS mode near the wall, leaving a "grey" area in between. The grey area is referred to as the regions modelled using an undetermined approach somewhere between RANS and LES.

Referring to the type B grid, inside the boundary layer (very close to the wall), the DES model switches to LES mode ($l_{DES} = C_{DES}\Delta$). However, this "ambiguous" grid is not fine enough to be able to resolve velocity fluctuations of LES and therefore lowers the eddy viscosity, which in turns results in modelled stress depletion (MSD).

The grid dependent GIS and MSD problems have a negative impact when it comes to a wide range of applications, resulting in inappropriate grid design parameters as it is very difficult/ infeasible to maintain a sufficiently coarse grid resolution tangential to the wall. Additionally, the fact that a refined grid having a negative impact on the computational results, is very contradictory and unusual. Therefore, there has been various methods proposed by researchers in order to

alleviate this problem, for instance, improvements to the original DES formulation, i.e. the delayed detached-eddy simulation (DDES) [57] and the improved delayed detached-eddy simulation (IDDES) [57] approaches.

7.3.1 Overcoming the limitations of the DES approach

The limitations of DES have been attempted to be solved by the development of enhanced versions of the DES models. In order to overcome the problem of the DES model introducing the RANS damping terms in LES mode, a correction function was proposed by Spalart et al. [74]. This correction function, denoted by Ψ is incorporated to the subgrid length scale l_{LES}, which can be represented by

$$l_{LES} = \Psi C_{DES} \Delta \tag{7.8}$$

This correction function, Ψ, which is dependant on the eddy viscosity ratio, operates by restoring the characteristics of a Smagorinsky-like model, and diverges from the value of 1 when the subgrid viscosity is relatively low. For RANS models not containing any low Reynolds number parameter (for example, the $k-\omega$ model of Wilcox [141]), $\Psi = 1$. A complete description of this correction function, together with its mathematical formulation are available in [74].

Additionally, in order to maintain the RANS mode functionality in the boundary layer regions, the MSD problem needs to be resolved. The first solution towards overcoming the MSD problem in particular, was proposed by Menter and Kuntz [142]. They suggested to employ the blending functions, F_1 and F_2 from Menter's SST model [67] to alleviate the problem of MSD, by identifying the boundary layer region and restricting the DES model to prematurely switch into LES mode within that region. The delayed detached-eddy simulation, DDES approach, derived from the initial effort of Menter and Kuntz [142] was proposed by Spalart et al. [74]. DDES is a generalised approach that was developed to be compatible with any background RANS model. To overcome the GIS problem, the proposed DDES model [74] switches the SGS formulation in the SA model as described by the boundary layer sensor function r_d, the shield function f_d and the modified DES length scale definition, l_{DDES},

$$r_d = \frac{\nu + \nu_t}{\kappa^2 {d_w}^2 \sqrt{\frac{\partial U_i}{\partial x_j}\frac{\partial U_i}{\partial x_j}}} \tag{7.9}$$

$$f_d = 1 - \tanh\left[(8 r_d)^3\right] \tag{7.10}$$

$$l_{DDES} = d_w - f_d \max\left(0, d_w, -C_{DES}\Delta\right) \tag{7.11}$$

The boundary layer sensor function r_d is simply a modification of the r function of the SA model as described in equation (7.4) in order to be applicable to any eddy viscosity background RANS model used within a DES formulation. The tanh blending function within the shield function f_d, ensures that the initial switch of the DES model to the LES mode is effective just outside the boundary layer, thereby addressing the grey area problem of DES ($f_d = 0$ inside the boundary layer and smoothly blends to $f_d = 1$ at the outer edge of the boundary layer). The primary role of f_d in the modified definition of the DES length scale, equation (7.11), is to delay the DES model to switch to LES mode, until the outside edge of the boundary layer is reached and hence the name delayed detached-eddy simulation (DDES).

Other improvements of the DES approach has been proposed, for instance, the improved delayed detached-eddy simulation (IDDES). Since the IDDES is not within the scope of this work, no

further elaboration on IDDES will be made. The interested reader is referred to [68] for a detailed background on the IDDES approach, together with the mathematical formulation.

7.4 Formulation and implementation of the proposed hybrid RANS/LES model

This section describes the mathematical formulation and implementation in OpenFOAM 2.3.x of the proposed hybrid RANS/LES model for transitional boundary layer applications. The proposed hybrid RANS/LES model is developed based on the DES formulation strategy of Strelets [66], which is essentially an extension of the original DES approach of Spalart et al. [1] for its applicability to two equation models. Strelets [66] presented a DES as well as a DDES model based on Menter's [67] $k-\omega$ SST model. The hybridisation methodology involved a simple modification, by substituting the length scale, $l_{k-\omega}$ within the destruction term of the k transport equation of the original SST RANS model by the DES length scale, which is defined by $l_{DES} = \min\{l_{k-\omega}, C_{DES}\Delta\}$. Using the hybridisation approach of Strelets [66], the two hybrid RANS/LES models will be formulated, which will be referred to as the $\boldsymbol{k_T - k_L - \omega}$ **DES** and $\boldsymbol{k_T - k_L - \omega}$ **DDES** and will be assigned the same names when implemented in OpenFOAM.

7.4.1 $k_T - k_L - \omega$ DES model

The modified version of the $k_T - k_L - \omega$ model [7] was employed as the background RANS model for the formulation of the proposed model, as described in section 3.3.4.4. The RANS model transport equations are described as follows: the turbulent kinetic energy k_T, laminar kinetic energy k_L and specific dissipation rate ω respectively.

$$\frac{Dk_T}{Dt} = P_{k_T} + R_{BP} + R_{NAT} - \omega k_T - D_T + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\alpha_T}{\sigma_k}\right)\frac{\partial k_T}{\partial dx_j}\right] \tag{7.12}$$

$$\frac{Dk_L}{Dt} = P_{k_L} - R_{BP} - R_{NAT} - D_L + \frac{\partial}{\partial x_j}\left[\nu\frac{\partial k_L}{\partial dx_j}\right] \tag{7.13}$$

$$\frac{D\omega}{Dt} = C_{\omega_1}\frac{\omega}{k_T}P_{k_T} + \left(\frac{C_{\omega R}}{f_W} - 1\right)\frac{\omega}{k_T}(R_{BP} + R_{NAT}) - C_{\omega 2}\omega^2 + C_{\omega 3}f_\omega\alpha_T f_W^2\frac{\sqrt{k_T}}{d^3} + \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\rho\alpha_T}{\alpha_\omega}\right)\frac{\partial\omega}{\partial x_j}\right] \tag{7.14}$$

As stated earlier, the DES formulation involves the substitution of the RANS length scale with the DES length scale. In the case of this formulation, taking into account the DES methodology of Strelets [66], the modification is carried out through the turbulent kinetic energy k_T transport equation.

The destruction term in the turbulent kinetic energy equation k_T can be represented by *epsilon* and is defined as

$$\epsilon = \omega k_T \tag{7.15}$$

The effective length scale, λ_{eff}, also referred to as the wall limited turbulence length scale, is given by

$$\lambda_{eff} = \min\left(C_\lambda d, \lambda_T\right) \tag{7.16}$$

where λ_T is the length scale, given by

$$\lambda_T = \frac{\sqrt{k_T}}{\omega} \tag{7.17}$$

The function of this DES formulation is to switch from the $k_T - k_L - \omega$ RANS model to an SGS model where the turbulent length scale, λ_T, predicted by the RANS model is larger than the local grid size, Δ. In this case, the length scale used for the calculation of the dissipation rate, ω in the k_T equation is substituted by the local grid size, Δ

The DES length scale l_{DES} is defined as,

$$l_{DES} = \min\{\lambda_T, C_{DES}\Delta\} \tag{7.18}$$

From equation (7.17) the dissipation rate is given by

$$\omega = \sqrt{k_T}/\lambda_T \tag{7.19}$$

Applying the DES condition by substituting λ_T by l_{DES}

$$\omega = \sqrt{k_T}/\lambda_T \qquad \rightarrow \qquad \sqrt{k_T}/\left(C_{DES}\Delta\right) \qquad \textit{for} \qquad \left(C_{DES}\Delta < \lambda_T\right) \tag{7.20}$$

where Δ is the maximum length of the local grid size, given by

$$\Delta = \max\{\Delta_x, \Delta_y, \Delta_z\} \tag{7.21}$$

Δ_x, Δ_y, Δ_z is the distance between the cell faces in the x, y and z directions and C_{DES} is the DES model constant

The DES modification is essentially a multiplier to the destruction term in the k_T transport equation as illustrated below

$$\frac{Dk_T}{Dt} = P_{k_T} + R_{BP} + R_{NAT} - F_{DES}\left(\omega k_T\right) - D_T + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\alpha_T}{\sigma_k}\right)\frac{\partial k_T}{\partial dx_j}\right] \tag{7.22}$$

where

$$F_{DES} = \max\left(\frac{\sqrt{k_T}}{C_{DES}\,\omega\Delta}, 1\right) \tag{7.23}$$

The laminar kinetic energy k_L and the energy dissipation ω model equations remain unchanged as shown in equation (7.13) and (7.14) respectively. The new hybrid RANS/LES model, i.e. the $k_T - k_L - \omega$ DES model was implemented in OpenFOAM 2.3.x, for which part of the source code is described as follows. The complete OpenFOAM source code of the $k_T - k_L - \omega$ DES model is described in Appendix C.

The F_{DES} term definition in OpenFOAM,

```
// F_DES term definition

tmp<volScalarField> kkLOmegaDES::FDES() const
{
    return max(scalar(1), sqrt(kt_)/(CDES_*omega_*delta()));
}
```

The turbulent kinetic energy k_T equation in OpenFOAM,

```
    {
        fvScalarMatrix ktEqn
        (
fvm::ddt(kt_)
      + fvm::div(phi_, kt_)
      - fvm::laplacian(DkEff(alphaTEff), kt_, "laplacian(alphaTEff,kt)")
     ==
Pkt
      + (Rbp + Rnat)*kl_
      - Dt
      - fvm::Sp(FDES*omega_, kt_)
);

ktEqn.relax();
ktEqn.solve();
    }
```

The energy dissipation rate ω equation in OpenFOAM,

```
    {
        fvScalarMatrix omegaEqn
        (
fvm::ddt(omega_)
      + fvm::div(phi_, omega_)
      - fvm::laplacian(DomegaEff(alphaTEff), omega_)
     ==
Cw1_*Pkt*omega_/(kt_ + kMin_)
      - fvm::SuSp
(
    (1.0 - CwR_/(fw + fwMin))*kl_*(Rbp + Rnat)/(kt_ + kMin_)
  , omega_
)
      - fvm::Sp(Cw2_*sqr(fw)*omega_, omega_)
      + Cw3_*fOmega(lambdaEff, lambdaT)*alphaTEff*sqr(fw)*sqrt(kt_)/pow3(y_)
     );

        omegaEqn.relax();
        omegaEqn.solve();
    }
```

The laminar kinetic energy k_L equation in OpenFOAM,

```
    {
        fvScalarMatrix klEqn
        (
fvm::ddt(kl_)
      + fvm::div(phi_, kl_)
      - fvm::laplacian(nu(), kl_)
     ==
Pkl
      - fvm::Sp(Rbp + Rnat + Dl/(kl_ + kMin_), kl_)
);

        klEqn.relax();
        klEqn.solve();
    }
```

7.4.1.1 How does the F_{DES} function operate?

When the local grid size is small enough in the x, y and z directions, in comparison to the length scale (RANS), $F_{DES} > 1$. This therefore destroys k, which further reduces ν_t (based on its definition in section 3.3.4.4. Hence, in regions where the grid is fine enough to resolve turbulence, the modelled turbulent eddy viscosity is minimised, allowing the DES model to switch to LES mode.

However, as discussed in section 7.3, the problem of GIS may arise, as the grid is generally very fine near the wall and there is a risk of the DES model switching to the LES mode, due to high MSD. This may result in badly resolved LES, since at the wall, the turbulent structures are very small. Therefore, as a result of this problem, the DDES approach was introduced. A DDES version of the $k_T - k_L - \omega$ DES model was also formulated and implemented in OpenFOAM (as described in section 7.4.2), in order to address the serious limitation of the DES approach.

7.4.1.2 The C_{DES} coefficient

C_{DES} is the model empirical constant. After having reviewed various DES models [1, 133, 134] for their C_{DES} coefficient (as illustrated in table 7.2), a value of 0.61 was assigned for the current DES model constant, which is within the range used in previous DES models.

Table 7.2: C_{DES} value used by previous DES models

SA DES	SAE DES	SALSA DES	WCX DES
0.65	0.65	0.60	0.70

The C_{DES} value assigned to the proposed DES and DDES models represents an initial estimate to allow the numerical tests to be carried out, primarily to verify the model implementation. This particular parameter needs to be calibrated in the future.

7.4.2 $k_T - k_L - \omega$ DDES model

The $k_T - k_L - \omega$ DDES model formulation involves the same development/implementation steps as those for the $k_T - k_L - \omega$ DES model formulation (as described in section 7.4.1) up to the definition of the F_{DES} function. For the DDES formulation, only the F_{DES} function of the $k_T - k_L - \omega$ DES model needs to be modified to incorporate the blending function of Menter [67] and is referred to as the F_{DDES} function.

The k_T equation of the $k_T - k_L - \omega$ RANS transition model, together with the incorporated DDES modification to the destruction term is given as:

$$\frac{Dk_T}{Dt} = P_{k_T} + R_{BP} + R_{NAT} - F_{DDES}\left(\omega k_T\right) - D_T + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\alpha_T}{\sigma_k}\right)\frac{\partial k_T}{\partial dx_j}\right] \tag{7.24}$$

where

$$F_{DDES} = \max\left(\frac{\sqrt{k_T}}{C_{DES}\,\omega\Delta}\left(1 - F_1\right), 1\right) \tag{7.25}$$

F_1 is the blending function used in Menter's $k-\omega$ SST model [67]. This blending function operates

by switching to $F_1 = 1$ inside the boundary layer and to 0 at the edge of the boundary layer.

$$F_1 = \tanh\left\{\left\{\min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right), \frac{4\sigma_{\omega 2}k}{CD_{k\omega}y^2}\right]\right\}^4\right\} \tag{7.26}$$

The $k_T - k_L - \omega$ DDES model was implemented in OpenFOAM by modifying the F_{DES} term from the $k_T - k_L - \omega$ DES model source code and defining the blending function F_1. Part of the OpenFOAM implementation of the DDES model is as follows. The complete source code of the $k_T - k_L - \omega$ DDES model is described in Appendix D.

The F_{DDES} term definition in OpenFOAM,

```
tmp<volScalarField> kkLOmegaDDES::FDDES(const volScalarField& FS) const
{
    return max
    (
     sqrt(kt_)/(CDES_*omega_*delta())*(scalar(1) - FS),
scalar(1)
    );
}

tmp<volScalarField> kkLOmegaDDES::F1(const volScalarField& CDkOmega) const
{
    tmp<volScalarField> CDkOmegaPlus = max
    (
        CDkOmega,
        dimensionedScalar("1.0e-10", dimless/sqr(dimTime), 1.0e-10)
    );

    tmp<volScalarField> arg1 = min
    (
        min
        (
            max
            (
                (sqrt(kt_)/(CmuStd_*omega_*y_)),
                scalar(500)*nu()/(sqr(y_)*omega_)
            ),

        (4*alphaOmega2_)*kt_/(CDkOmegaPlus*sqr(y_))
        ),
        scalar(10)
    );

    return tanh(pow4(arg1));
}
```

The turbulent kinetic energy k_T equation in OpenFOAM,

```
    {
        fvScalarMatrix ktEqn
        (
        fvm::ddt(kt_)
      + fvm::div(phi_, kt_)
      - fvm::laplacian(DkEff(alphaTEff), kt_, "laplacian(alphaTEff,kt)")
     ==
        Pkt
      + (Rbp + Rnat)*kl_
      - Dt
      - fvm::Sp(FDDES*omega_, kt_)
        );

        ktEqn.relax();
        ktEqn.solve();
    }
```

The energy dissipation rate ω equation in OpenFOAM,

```
    {
        fvScalarMatrix omegaEqn
        (
fvm::ddt(omega_)
      + fvm::div(phi_, omega_)
      - fvm::laplacian(DomegaEff(alphaTEff), omega_)
     ==
Cw1_*Pkt*omega_/(kt_ + kMin_)
      - fvm::SuSp
(
    (1.0 - CwR_/(fw + fwMin))*kl_*(Rbp + Rnat)/(kt_ + kMin_)
  , omega_
)
      - fvm::Sp(Cw2_*sqr(fw)*omega_, omega_)
      + Cw3_*fOmega(lambdaEff, lambdaT)*alphaTEff*sqr(fw)*sqrt(kt_)/pow3(y_)
        );

        omegaEqn.relax();
        omegaEqn.solve();
    }
```

The laminar kinetic energy k_L equation in OpenFOAM,

```
    {
        fvScalarMatrix klEqn
        (
fvm::ddt(kl_)
      + fvm::div(phi_, kl_)
      - fvm::laplacian(nu(), kl_)
     ==
Pkl
      - fvm::Sp(Rbp + Rnat + Dl/(kl_ + kMin_), kl_)
    );

        klEqn.relax();
        klEqn.solve();
    }
```

7.5 Numerical Testing

Following the implementation of the hybrid RANS/LES models, numerical tests were performed in order to verify whether the models were implemented correctly. These initial numerical tests are also intended to verify the correct behaviour of the new hybrid RANS/LES models. The numerical tests were performed on a 2D zero-pressure-gradient flat plate and 2D cylinder.

7.5.1 Numerical setup

The computational domain size used for the DES/DDES simulations was $2.95m \times 0.175m \times 0.025m$ in the streamwise, wall-normal and spanwise direction respectively with a plate length of $2.9m$. Figure 7.3 is a sketch representation of the computational domain and boundary names. The purpose of the region upstream of the plate was to define the correct inlet boundary condition for the laminar kinetic energy (i.e. a value of 0 at the inlet). The simulations were performed with a reasonably fine grid ($y^+ \sim 0.6$ at the wall). The 2D grid was generated using the blockMesh utility available in OpenFOAM, and consisted of a fully structured hexahedral grid with $640 \times 300 \times 1$ cells. The freestream velocity, U_∞ and turbulence intensity, Tu prescribed at the inlet are $5.4m/s$ and 3.0% respectively.

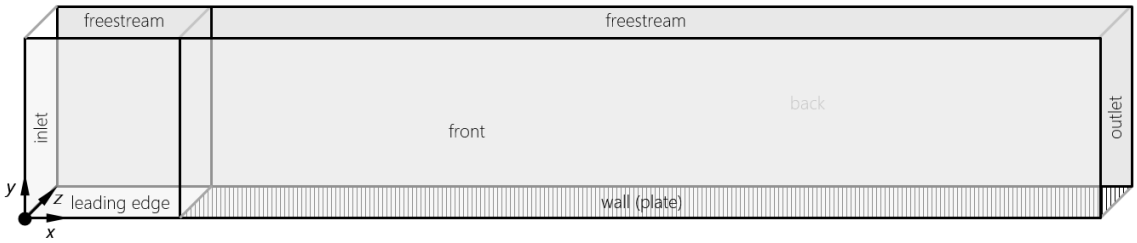


Figure 7.3: Sketch of the DES case geometry (2D flat plate)

The filtered Reynolds-averaged Navier-Stokes equations along with the transport equations of the DES/DDES model were initially solved using the incompressible, unsteady, turbulent PISO solver, readily implemented and available in OpenFOAM. A second order implicit backward Euler (also referred to as "backward" in OpenFOAM) time discretisation scheme was used and a central dicretisation scheme, *Gauss Linear* scheme was used to compute the gradient and the divergence terms. The turbulent variables were computed using a limited central discretisation scheme, *Gauss limitedLinear 1*. The pressure equation was solved using the geometric algebraic multi grid *GAMG* solver, with *FDIC* smoothing and a tolerance of 10^{-7} and a relative tolerance of 0 between iterations. The remaining equations were solved using the *smoothSolver* solver together with the diagonal incomplete LU *DILU* asymmetric preconditioner, with a tolerance of 10^{-9} and a relative tolerance of 0. In order to achieve a stable solution, each time-step, Δt was automatically adjusted at run-time to keep the Courant number, C_o, to be less than 1 (approximately 0.6).

7.5.2 Boundary conditions

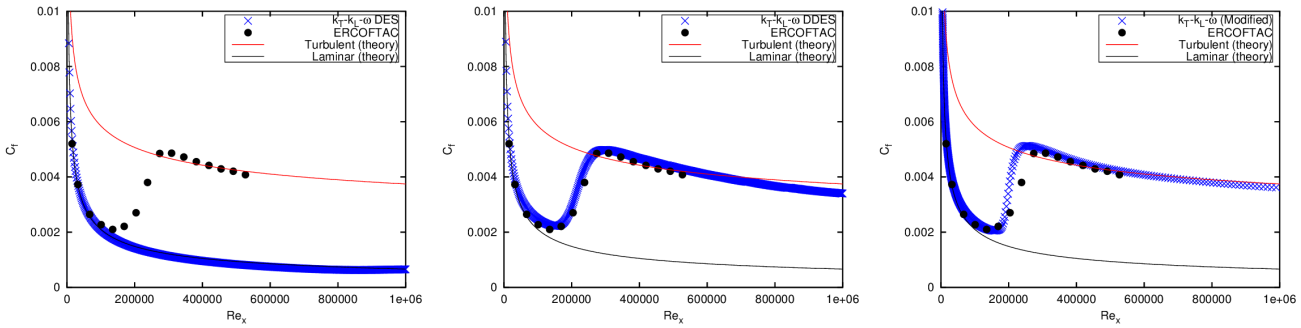
Since the $k_T - k_L - \omega$ DES and the $k_T - k_L - \omega$ DDES models are newly developed models, it is important to identify the appropriate boundary conditions. The boundary conditions prescribed for the new DES and DDES models were very similar to those prescribed for the RANS simulations using the $k_T - k_L - \omega$ transition model. The detailed boundary conditions for the 2D flat plate case are illustrated in table 7.3.

Table 7.3: Boundary conditions for flat plate computations, $k_T - k_L - \omega$ DES and $k_T - k_L - \omega$ DDES model

	Inlet	**Outlet**	**Wall**	**Front**	**Back**	**Freestream**	**Leading Edge**
$\boldsymbol{U}$	fixedValue (5.4 0 0)	pressureInletOutletVelocity tangentialVelocity (0 0 0)	fixedValue (0 0 0)	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{p}$	zeroGradient	fixedValue 0	zeroGradient	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{k_T}$	fixedValue 0.00394	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
$\boldsymbol{k_L}$	fixedValue 1×10^{-15}	zeroGradient	fixedValue 1×10^{-15}	empty	empty	zeroGradient	symmetryPlane
ω	fixedValue 2.498	zeroGradient	zeroGradient	empty	empty	zeroGradient	symmetryPlane
ν_{sgs}	calculated 0	calculated 0	calculated 0	empty	empty	calculated 0	symmetryPlane

7.5.3 Results & Discussion

Figure 7.4 illustrates the numerical test results of the $k_T - k_L - \omega$ DES model and the $k_T - k_L - \omega$ DDES model when tested on a 2D zero-pressure-gradient flat plate case. The plots represents the skin friction coefficient, C_f against the local Reynolds number, Re_x.



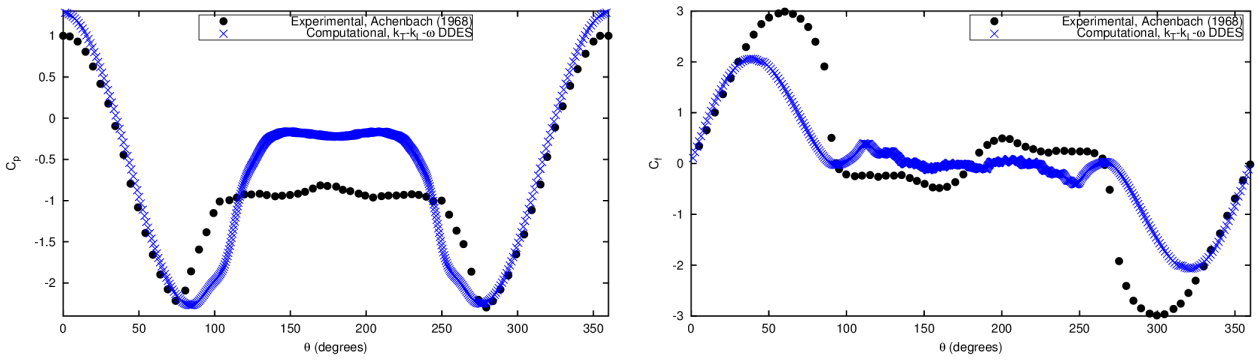
(a) $k_T - k_L - \omega$ DES **(b)** $k_T - k_L - \omega$ DDES **(c)** $k_T - k_L - \omega$

Figure 7.4: T3A, Tu = 3.0%, $k_T - k_L - \omega$ DES and $k_T - k_L - \omega$ DDES results

The results obtained from the $k_T - k_L - \omega$ DES model did not show good agreement with the experimental results, which can be due to various factors. As discussed earlier in section 7.3, the DES method is very sensitive to the near wall grid resolution. The computational results, as illustrated in figure 7.4a, demonstrated that the model struggled to capture transition to turbulence, for which can be due to the effect of MSD. To overcome this problem, the computational grid has to be carefully designed to prevent the DES model of switching to LES mode within the attached boundary layer region.

The $k_T - k_L - \omega$ DDES model, when tested on the same 2D flat plate case using the same computational grid, provided very good agreement with the experimental results (as illustrated in figure 7.4b). These results indicated that the implemented DDES version of the model addressed the limitations of the $k_T - k_L - \omega$ DES model. The DDES results, when compared with the results of the state-of-the-art RANS model, i.e. the $k_T - k_L - \omega$ transition model (figure 7.4c), demonstrated approximately a 30% improvement in the prediction of boundary layer transition location.

The $k_T - k_L - \omega$ DDES model was further tested on a 2D cylinder case [116] at $Re = 2.6 \times 10^5$. The results obtained are illustrated in figure 7.5, which represents the pressure coefficient, C_p and the skin friction coefficient C_f against the angle θ around the cylinder.



(a) C_p v/s θ **(b)** C_f v/s θ

Figure 7.5: 2D cylinder, $k_T - k_L - \omega$ DDES results

The freestream velocity and freestream pressure boundary conditions were prescribed at the inlet and outlet, using an o-grid respectively. The turbulent quantities were computed based on the model equations and a freestream condition was prescribed. At the wall, a fixed value (Dirichlet) boundary condition was prescribed for k_L and k_T and a zeroGradient condition was imposed on ω. Although the results, as illustrated in figure 7.5, do not completely match the experimental results, they show a correct trend. The possible reason(s) for which the computational results do not closely match the experimental results may be because the simulation was performed in 2D and/or that the model needs to be calibrated since it is a newly implemented model.

The hybrid RANS/LES models, when tested on simple 2D configurations, i.e. 2D zero-pressure-gradient flat plate and 2D cylinder, the computational results achieved clearly demonstrated that the models were implemented correctly in OpenFOAM and that they are fully operational. However, the two DES models developed as part of this work require further testing developed DES models require further testing for calibration purposes to validate/assess their applicability.

Chapter 8

Conclusion

A novel hybrid RANS/LES model for transitional boundary layers has been proposed and developed. The hybridisation approach was based on the detached-eddy simulation (DES) modelling framework of Spalart et al. [1], using the $k_T - k_L - \omega$ transition model of Walters and Colkjat [2] as the background RANS model and the model was implemented in OpenFOAM 2.3.x. In fact, two versions of the hybrid RANS/LES were implemented as part of this current work: the $k_T - k_L - \omega$ DES model and the $k_T - k_L - \omega$ DDES model.

The first model, i.e. the $k_T - k_L - \omega$ DES, when numerically tested on a 2D zero-pressure-gradient flat plate demonstrated to be fully functional. However, the results achieved from the numerical test indicated that the the $k_T - k_L - \omega$ DES model failed to capture laminar to turbulent transition and the computational solution remained laminar. This particular behaviour of the DES model was due to its high sensitivity to the computational grid. The DES method tends to suffer from the problem of MSD, which occurs as a result of the high grid resolution near the wall, which in turns forces the DES model to switch to LES mode in the near-wall region, within the boundary layer. Therefore, a delayed detached-eddy simulation version (DDES) of the model, i.e. the $k_T - k_L - \omega$ DDES model was developed to address the problem of MSD and GIS. The $k_T - k_L - \omega$ DDES model when tested on a 2D zero-pressure-gradient flat plate showed excellent agreement when compared to the experimental skin friction distribution of the ERCOFTAC T3A case (3.0% freestream turbulence intensity) [3]. In fact, the results of the $k_T - k_L - \omega$ DDES model, when compared to the $k_T - k_L - \omega$ RANS transition model [2] showed an improvement of approximately 30% in the prediction of the boundary layer transition location.

Furthermore, the $k_T - k_L - \omega$ DDES model features a series of positive attributes. Unlike LES, the $k_T - k_L - \omega$ DDES model does not require the flow to be perturbed at the inlet. Therefore, the definition of inlet boundary conditions is simple compared to pure LES, since the model does not require complex inlet boundary conditions in order to predict transition. The boundary conditions prescribed for the $k_T - k_L - \omega$ DDES are identical to the $k_T - k_L - \omega$ RANS transition model of Walters and Cokljat [2]. The $k_T - k_L - \omega$ DDES model was also able to capture/predict boundary layer transition when using a 2D configuration, as opposed to pure LES which can only capture transition exclusively on 3D grids. Additionally, an important advantage of the $k_T - k_L - \omega$ DDES model is that it requires significantly lower grid density compared to pure LES. The 2D flat plate DES simulations used a grid of $650 \times 300 \times 1$ compared to the 2D LES computations, which used a grid of $630 \times 515 \times 1$ and still LES failed to capture transition in 2D. This represents a 40% lower grid resolution required by the $k_T - k_L - \omega$ DDES model compared to 2D LES.

The hybrid RANS/LES models were implemented in OpenFOAM, because it is open source, and it grants the user full access to the source code, providing the user with the flexibility to modify the code. However, OpenFOAM has not yet been widely adopted by industry. For this reason and since the $k_T - k_L - \omega$ DDES is a new model (requiring comprehensive validation), it cannot be readily considered as an industrial tool. However, the model will definitely benefit the open source and research community.

8.1 Contributions

The key contributions are summarised as follows:

- The implementation of the SIMPLEC algorithm in OpenFOAM 2.3.x (also compatible with version 2.4.x).
- Documented and validated several RANS turbulence models readily available in OpenFOAM, list of models.
- Assessment of various SGS models available in OpenFOAM.
- Assessment of various inflow conditions for LES, in order to identify suitable inlet conditions for boundary layer transition.
- Formulation and implementation of the $k_T - k_L - \omega$ DES and $k_T - k_L - \omega$ DDES models, in OpenFOAM 2.3.x.
- Contribution to the open source community: two novel hybrid RANS/LES models implemented in OpenFOAM.
- Invited conference paper for proceedings at the Royal Aeronautical Society, General Aviation Conference, London, November 2015 [143].

8.2 Future work

This work also identified several areas that deserve further study. Some suggestions include:

- Further investigations on LES inlet boundary conditions since the LES inflow conditions assessment carried out during this research was only exploratory. The definition and development of suitable inflow conditions for large-eddy simulation is generally considered to be major area of research in its own right.
- Calibration of the implemented hybrid RANS/LES models, by testing a range of different values of the model constant C_{DES} on various configurations, in order to identify a suitable value for C_{DES}, which can be generalised.
- The implemented models can be tested for different available spatial filters, Δ in OpenFOAM. In this work, only the $\Delta = \max\{\Delta_x, \Delta_y, \Delta_z\}$ filter was tested.
- The implemented hybrid RANS/LES models can be validated on an aerofoil case and other 3D configurations.
- An assessment of various boundary conditions, available in OpenFOAM, on the implemented models can be carried out, in order to assess their influence on the behaviour of the models.

- Formulation and implementation of the IDDES version of the $k_T - k_L - \omega$ DES model in OpenFOAM.

References

[1] P.R. Spalart. Comments on the feasibility of LES for wings, and on a hybrid RANS/LES approach. In *Proceedings of the 1st AFOSR International Conference on DNS/LES, Louisiana Tech University, USA, August 4-8, 1997*, 1997.

[2] D.K. Walters and D. Cokljat. A three-equation eddy-viscosity model for Reynolds-averaged Navier-Stokes simulations of transitional flow. *Journal of Fluids Engineering*, 130(12):1–14, 2008.

[3] J. Coupland. ERCOFTAC special interest group on laminar to turbulent transition and retransition: T3A and T3B. Technical Report A309514, ERCOFTAC, 1990.

[4] J.L. van Ingen. The e^N method for transition prediction. Historical review work at TU Delft. In *38th AIAA Fluid Dynamics Conference and Exhibit*, 2008.

[5] F. Hachem and M.W. Johnson. Boundary layer transition correlation for concave surfaces. In *ASME 35th International Gas Turbine and Aeroengine Congress and Exposition*, 1990.

[6] J.A. Masad and M.R. Malik. Transition correlation in subsonic flow over a flat plate. *AIAA Journal*, 31(10):1953–1955, 1993.

[7] H. Medina and J. Early. Modelling transition due to backward-facing steps using the laminar kinetic energy concept. *European Journal of Mechanics B/Fluids*, 44:60–68, 2014.

[8] U. Piomelli. Large-eddy simulation: achievements and challenges. *Progress in Aerospace Sciences*, 35(4):335–362, 1999.

[9] D. Meyer, U. Rist, and S. Wagner. DNS of the generation of secondary vortices in a transitional boundary layer. *Advances in Turbulence VII. Proceedings of the Seventh European Turbulence Conference*, pages 97 – 100, 1998.

[10] P. Schlatter. Large-eddy simulation of transition and turbulence in wall-bounded shear flow. Technical Report PhD Thesis, Swiss Federal Institute of Technology Zürich, 2005.

[11] H. Fasel. Investigation of the stability of boundary layers by a finite-difference model of the Navier-Stokes equations. *Journal of Fluid Mechanics*, 78:335–383, 1976.

[12] H. Fasel and H. Bestek. Investigation of non-linear, spatial disturbance amplification in plane Poiseuille flow. In R. Eppler and H. Fasel, editors, *Laminar-Turbulent Transition: IUTAM symposium, Stuttgart, Germany, September 16-22, 1979*, pages 173–185. Springer-Verlag, 1980.

[13] N. Gilbert. Numerische simulationn der transition von der laminaren in die turbulent kanalströmung. Technical Report PhD Thesis, Institut für theoretische Strömungsmechanik, Germany, 1988.

[14] N. Gilbert and L. Kleiser. Near wall phenomena in transition to turbulence. In S.J. Kline and N.H. Afgan, editors, *Near-Wall Turbulence - 1988 Zoran Zarić Memorial Conference*, pages 7–27, Hemisphere, New York, USA, 1990.

[15] Lin Chen, Xiaobing Liu, Maria Oliveira, and Chaoqun Liu. DNS for late stage structure of flow transition on a flat-plate boundary layer. *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010.

[16] Chaoqun Liu and Ping Lu. DNS study on physics of late boundary layer transition. *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2012.

[17] Ping Lu and Chaoqun Liu. DNS study on mechanism of small length scale generation in late boundary layer transition. *Physica D*, 241(1):11 – 24, 2012.

[18] M.M. Rai and P. Moin. Direct numerical simulation of transition and turbulence in a spatially evolving boundary layer. *Journal of Computational Physics*, 109:169–192, 1993.

[19] T. Sayadi, C.W. Hamman, and P. Moin. Direct numerical simulation of K-type and H-type transitions to turbulence. *Centre of Turbulence Research Annual Research Brief*, pages 109–121, 2011.

[20] Y.S. Kachanov and V.Y. Levchenko. The resonant interaction of disturbances at laminar-turbulent transition in a boundary layer. *Journal of Fluid Mechanics*, 12:1–34, 1984.

[21] U. Rist and H. Fasel. Direct numerical simulation of controlled transition in a flat-plate boundary layer. *Journal of Fluid Mechanics*, 298:211–248, 1995.

[22] W. Zhang, R. Hain, and C.J. Kahler. Scanning PIV investigation of the laminar separation bubble on a SD7003 airfoil. *Experiments in Fluids*, 45:725–743, 2008.

[23] U. Gaitonde. Quality criteria for large-eddy simulation. Technical Report PhD Thesis, The University of Manchester, 2008.

[24] X. Zheng, C. Liu, F. Liu, and C. Yang. Turbulent transition simulation using the $k-\omega$ model. *International Journal of Numerical Methods in Engineering*, 42:907–926, 1998.

[25] X. Wu and P. Moin. Direct numerical simulation of turbulence in a nominally zero-pressure-gradient flat-plate boundary layer. *Journal of Fluid Mechanics*, 630:5–41, 2009.

[26] I. Hadzic and K. Hanjalic. Separation-induced transition to turbulence: second-moment closure modelling. *Flow, Turbulence and Combustion*, 63(1):153–173, 1999.

[27] W.P. Jones and B.E. Launder. The prediction of laminarization with a two-equation model of turbulence. *International Journal of Heat and Mass Transfer*, 15(2):301–314, 1972.

[28] D.D. Pasquale, A. Rona, and S.J. Garrett. A selective review of CFD transition models. In *Proceedings of the 39th AIAA Fluid Dynamics Conference, San Antonio, Texas, June 22-25, 2009*, 2009.

[29] A.M. Savill. Some recent progress in the turbulence modelling of by-pass transition. In R.M.C. So, C.G. Speziale, and B.E. Launder, editors, *Proceedings of the International Conference on Near-Wall Turbulent Flows*, pages 829–848. Elsevier, 1993.

[30] A.M. Savill. By-pass transition using conventional closures. *Closures Strategies for Turbulent and Transitional Flows*, pages 462–492, 2002.

[31] D.C. Wilcox. Simulation of transition with a two-equation turbulence model. *AIAA Journal*, 32(2):247–255, 1994.

[32] C.L. Rumsey. Apparent transition behaviour of widely-used turbulence models. In *Proceedings of the 36th AIAA Fluid Dynamics Conference, San Francisco, California, June 5-8, 2006*, pages 2625–2643, 2006.

[33] R. Langtry and F. Menter. Overview of industrial transition modelling in CFX. Technical Report TPL 8126, ANSYS, 2006.

[34] S. Fu and L. Wang. Modelling the flow transition in supersonic boundary layer with a new $k-\omega-\gamma$ transition/turbulence model. In *7th International Symposium on Engineering turbulence Modelling and Measurements-ETMM7, Limassol, Cyprus, June 4-6, 2008*, 2008.

[35] D.K. Walters and J.H. Leylek. A new model for boundary layer transition using a single-point RANS approach. *Transactions of the ASME. The Journal of Turbomachinery*, 126(1):193 – 202, 2004.

[36] B.J. Abu-Ghannam and R. Shaw. Natural transition of boundary layers - the effects of turbulence, pressure gradient, and flow history. *Journal of Mechanical Engineering Science*, 22(5):213 – 228, 1980.

[37] Y.B. Suzen and P.G. Huang. Modeling of flow transition using an intermittency transport equation. *Transactions of the ASME. Journal of Fluids Engineering*, 122(2):273 – 84, 2000.

[38] F.R. Menter, R.B. Langtry, S.R. Likki, Y.B. Suzen, P.G. Huang, and S. Volker. A correlation-based transition model using local variables-part I: model formulation. *Transactions of the ASME. The Journal of Turbomachinery*, 128(3):413 – 22, 2006.

[39] D.K. Walters and J.H. Leylek. Computational fluid dynamics study of wake-induced transition on a compressor-like flat plate. *Transactions of the ASME. The Journal of Turbomachinery*, 127(1):52 – 63, 2005.

[40] F. El-Salem. Triblade wind turbine - CFD simulation. Technical Report MSc Thesis, Lund University, 2015.

[41] J. Fürst. Numerical simulation of transitional flows with laminar kinetic energy. *Engineering Mechanics*, 20:379–388, 2013.

[42] J. Fürst, J. Příhoda, and P. Straka. Numerical simulation of transitional flows. *Computing*, 95(1):163–182, 2013.

[43] B. Shome. Numerical study of oscillating boundary layer flow over a flat plate using $k-k_L-\omega$ turbulence model. *International Journal of Heat and Fluid Flow*, 42:131 – 138, 2013.

[44] B. Shome. Numerical study of a flat-crested oscillatory boundary layer flow over a flat plate. *Computers & Fluids*, 92:151 – 159, 2014.

[45] M. Breuer, N. Jovičić, and K. Mazaev. Comparison of DES, RANS and LES for the separated flow around a flat plate at high incidence. *International Journal for Numerical Methods in Fluids*, 41(4):357–388, 2003.

[46] F. Peneau, H.C. Boisson, and N. Djilali. Large-eddy simulation of the influence of high freestream turbulence on a spatially evolving boundary layer. *International Journal of Heat and Fluid Flow*, 21:640–647, 2000.

[47] J. Smagorinsky. General circulation experiments with the primitive equations. *Monthly Weather Review*, 91(3):99–164, 1963.

[48] J. W. Deardorff. A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *Journal of Fluid Mechanics*, 41:453–480, 1970.

[49] U. Piomelli, T.A. Zang, C.G. Speziale, and M.Y. Hussaini. On the large-eddy simulation of transitional wall-bounded flows. *Physics of Fluids A*, 2(2):257–265, 1990.

[50] U. Piomelli, T.A. Zang, C.G. Speziale, and M.Y. Hussaini. On the large-eddy simulation of transitional wall-bounded flows. *Computer Physics Communications*, 65:224, 1991.

[51] Z. Yang, P.R. Voke, and A.M. Savill. Mechanisms and models of boundary layer receptivity deduced from large-eddy simulation of by-pass transition. In Voke et al., editor, *Direct and Large-Eddy Simulation I*, volume 26 of *Fluid Mechanics and Its Applications*, pages 225–236. Springer Netherlands, 1994.

[52] M. Germano, U. Piomelli, Moin P., and W.H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A*, 3(7):1760–1765, 1991.

[53] Peter R. Voke and Zhiyin Yang. Numerical study of bypass transition. *Physics of Fluids*, 7(9):2256–2264, 1995.

[54] X. Huai, R.D. Joslin, and U. Piomelli. Large-eddy simulation of transition to turbulence in boundary layers. *Theoretical Computational Fluid Dynamics*, 9:149–163, 1997.

[55] T. Sayadi and P. Moin. A comparative study of subgrid scale models for the prediction of transition in turbulent boundary layers. *Centre of Turbulence Research Annual Research Brief*, pages 237–247, 2010.

[56] T. Sayadi and P. Moin. Predicting natural transition using large eddy simulation. *Centre of Turbulence Research Annual Research Brief*, pages 97–108, 2011.

[57] P.R. Spalart. Strategies for turbulence modelling and simulations. *International Journal of Heat and Fluid Flow*, 21(3):252–263, 2000.

[58] A. Spille-Kohoff and H.J. Kaltenhach. Generation of turbulent inflow data with a prescribed shear-stress profile. Technical Report ADP013648, DTIC, 2011.

[59] J.U. Schluter, H. Pitsch, and P. Moin. Large-eddy simulation inflow conditions for coupling with Reynolds-averaged flow solvers. *AIAA Journal*, 42(3):478–484, 2004.

[60] P.R. Spalart and Allmaras S.R. A one-equation turbulence model for aerodynamic flows. In *AIAA92-0439, AIAA 30th Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, January 6-9, 1992*. AIAA, 1992.

[61] L. Davidson and S.H. Peng. Hybrid LES-RANS modelling: a one-equation SGS model combined with a $k - \omega$ model for predicting recirculating flows. *International Journal for Numerical Methods in Fluids*, 43(9):1003–1018, 2003.

[62] B. Zhong and P.G. Tucker. $k - l$ based hybrid LES/RANS approach and its application to heat transfer simulation. *International Journal for Numerical Methods in Fluids*, 46(10):983–1005, 2004.

[63] P.G. Tucker and L. Davidson. Zonal $k - l$ based large eddy simulations. *Computers and Fluids*, 33(2):267–287, 2004.

[64] L. Temmerman, M. Hadžiabdić, M.A. Leschziner, and K. Hanjalić. A hybrid two-layer URANS-LES approach for large eddy simulation at high Reynolds numbers. *International Journal of Heat and Fluid Flow*, 26(2):173–190, 2005.

[65] S. Jee and K. Shariff. Detached-eddy simulation based on the $v^2 - f$ model. *International Journal of Heat and Fluid Flow*, 46:84–101, 2014.

[66] M. Strelets. Detached-eddy simulation of massively separated flows. In *American Institute of Aeronautics & Astronautics, Washington, DC, AIAA-2001-0879*, 2001.

[67] F.R. Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, 32:1598–1605, 1994.

[68] W. Haase, M. Braza, and A. Revell. *DESider-A European Effort on Hybrid RANS-LES Modelling: Results of the European-Union Funded Project, 2004-2007*. Notes on Numerical Fluid Mechanics and Multidisciplinary Design. Springer, 2009.

[69] N.N. Sørensen, A. Bechmann, and F. Zahle. 3D CFD computations of transitional flows using DES and a correlation based transition model. *Wind Energy*, 14(1):77–90, 2011.

[70] F.R. Menter, R.B. Langtry, S.R. Likki, Y.B. Suzen, P.G. Huang, and S. Völker. A correlation-based transition model using local variables - Part I: Model formulation. *Journal of Turbomachinery*, 128(3):413–422, 2006.

[71] R.B. Langtry, F.R. Menter, S.R. Likki, Y.B. Suzen, P.G. Huang, and S. Völker. A correlation-based transition model using local variables - Part II: Test cases and industrial applications. *Journal of Turbomachinery*, 128(3):423–434, 2006.

[72] M.F. Alam, D.K. Walters, and D.S. Thompson. A transition-sensitive hybrid RANS/LES modeling methodology for CFD applications. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, Grapevine, Texas, January 7-10, 2013*, 2008.

[73] J.P. Boris, F.F. Grinstein, E.S. Oran, and R.L. Kolbe. New insights into large eddy simulation. *Fluid Dynamics Research*, 10(4-6):199, 1992.

[74] P.R. Spalart, S. Deck, M.L. Shur, K.D. Squires, M.Kh. Strelets, and A. Travin. A new version of Detached-eddy simulation, resistant to ambiguous grid densities. *Theoretical and Computational Fluid Dynamics*, 20(3):181–195, 2006.

[75] P. Bradshaw. *An Introduction to turbulence and its measurements*. Pergamon Press, 1985.

[76] L. Prandtl. Motions of fluids with very little viscosity. Technical Report 452, NACA, 1928.

[77] H. Blasius. Grenzschichten in Flüssigkeiten mit kleiner Reibung. *Z. Math. Phys.*, 56:1–37, 1908.

[78] H.L. Dryden. Air flow in the boundary layer near a plate. Technical Report 562, NACA, 1936.

[79] O. Reynolds. An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels. *Philosophical Transactions of the Royal Society of London*, 174:935–982, 1883.

[80] Boeing. Surface coating and drag reduction. *Boeing Aero Magazine*, 49(1):15–20, 2013.

[81] R.E. Mayle. Role of laminar-turbulent transition in gas turbine engines. *American Society of Mechanical Engineers*, 113(4):509–536, 1991.

[82] H. Schlichting. *Boundary layer theory: Seventh edition.* McGraw-Hill, 1979.

[83] J.T. Stuart. Stability and transition: Some comments on the problems. In R. Eppler and H. Fasel, editors, *Laminar-Turbulent Transition: IUTAM symposium, Stuttgart, Germany, September 16-22, 1979*, pages 1–13. Springer-Verlag, 1980.

[84] S.S. Saric, H. Reed, and E.J. Kerschen. Boundary layer receptivity to freestream disturbances. *Annual Review of Fluid Mechanics*, 34:291–319, 2002.

[85] M.S. Genç, I. Karasu, H.H. Açikel, and M.T. Akpolat. Low Reynolds number flows and transition. In M.S. Genç, editor, *Low Reynolds Number Aerodynamics and Transition*. InTech, 2012.

[86] S. Tavoularis. Boundary layer transition. University Lecture, 2014.

[87] M.V. Morkovin. On the many faces of transition. In C.S. Wells, editor, *Viscous Drag Reduction*, pages 1–31. Springer US, 1969.

[88] L.E. Jones, R.D. Sandberg, and N.D. Sandham. Direct numerical simulations of forced and unforced separation bubbles on an aerofoil at incidence. *Journal of Fluid Mechanics*, 602:175–207, 2008.

[89] M. Alam and N.D. Sandham. Simulations of bypass transition. *Journal of Fluid Mechanics*, 410:1–28, 2000.

[90] E. Witold. Transition modelling in turbomachinery. *Journal of Theoretical and Applied Mechanics*, 45(3):539–556, 2007.

[91] A.L. Braslow. A review of factors affecting boundary-layer transition. Technical Report TN D-3384, NASA Langley Research Center, 1966.

[92] M. Jahanmiri. Boundary layer transition in gas turbines. Technical Report 1, Chalmers University of Technology, 2011.

[93] Roelke R.J. and J.E. Haas. The effect of rotor blades thickness and surface finish on the performance of a small axial flow turbine. *Journal of Engineering for Gas Turbines and Power*, 105(2):377–382, 1983.

[94] J. Ferziger and M Perić. *Computational Methods for Fluid Dynamics*. Springer-Verlag, 2002.

[95] S.B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[96] J. Boussinesq. Essai sur la théorie des eaux courantes. *Mémoires présentés par divers savants à l'Académie des Sciences*, 23(1):1–680, 1872.

[97] F.G. Schmitt. About Boussinesq's turbulent viscosity hypothesis: Historical remarks and a direct evaluation of its validity. *Comptes Rendus Mécanique*, 335(9-10):617–627, 2007.

[98] A. Dewan. *Tackling Turbulent Flows in Engineering*. Springer-Verlag, 2011.

[99] L. Prandtl. Neuere ergebnisse der turbulenzforschung. *Zeitschrift des Vereins Deutscher Ingenieure VDI-Z*, 77(5):105–114, 1933.

[100] T. von Kármán. Mechanische Ähnlichkeit und turbulenz. In *Proceedings of the 3rd International Congress on Applied Mechanics, Stockholm, Sweden, Part 1, 1930*, page 85, 1930.

[101] E. de Villiers. The potential of Large-eddy simulation for modelling wall bounded flows. Technical Report PhD Thesis, Imperial College, 2006.

[102] P. Segaut and C. Méneveau. *Large Eddy simulation for incompressible flows: An Introduction*. Springer Verlag, 2006.

[103] D.K. Lilly. The representation of small scale turbulence in numerical simulation experiments. In *IBM Scientific Computing Symposium on environmental sciences*, pages 195–210, Yorktown heights, 1967.

[104] C. Mockett. *A Comprehensive Study of Detached Eddy Simulation*. Univ.-Verlag der TU, Univ.-Bibliothek, 2009.

[105] A. Yoshizawa and K. Horiuti. A statistically-derived subgrid-scale kinetic energy model for large-eddy simulation of turbulent flows. *Journal fo Physical Society of Japan*, 54(8):2834–2839, 1985.

[106] C. Fureby, A.D. Gosman, G. Tabor, H.G. Weller, N. Sandham, and M. Wolfshtein. Large eddy simulation of turbulent channel flows. In *11th Symposium on Turbulent Shear Flows, Grenoble, France*, pages 13–28, 1997.

[107] D.K. Lilly. A proposed modification of the Germano subgrid scale closure method. *Physics of Fluids A*, 4(3):633–635, 1992.

[108] U. Piomelli and J. Liu. Large-eddy simulation of rotating channel flows using a localized dynamic model. *Physics of Fluids*, 7(4):839–848, 1995.

[109] S. Ghosal, T.S. Lund, P. Moin, and A.K. A dynamic localization model for large-eddy simulation of turbulent flows. *Journal of Fluid Mechanics*, 286:229–255, 3 1995.

[110] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Taylor & Francis, 1980.

[111] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics. The Finite Volume Method*. Pearson Education Limited, 1997.

[112] D. Cappelli and N.N. Mansour. Performance of Reynolds-averaged Navier-Stokes models in predicting separated flows: Study of the hump flow model problem. In *Proceedings of the 36th AIAA Applied Aerodynamics Conference, San Diego, California, June 24-27, 2013*, pages 1–26, 2013.

[113] J.P. Van Doormaal and G.D. Raithby. Enhancements of the SIMPLE method for predicting incompressible fluid flows. *Numerical Heat Transfer*, 7(2):147–163, 1984.

[114] R.I. Issa. Solution of the implicitly discretized fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62:40–65, 1985.

[115] J. Tu, G.H. Yeoh, and C. Liu. *Computational Fluid Dynamics - A Practical Approach*. Butterworth-Heinemann, 2008.

[116] E. Achenbach. Distribution of local pressure and skin friction around a circular cylinder in cross-flow up to $re = 5 \times 10^6$. *Journal of Fluid Mechanics*, 34:625–639, 12 1968.

[117] G.C. Speziale, R. Abid, and E.C. Anderson. A critical evaluation of two-equation models for near wall turbulence. *AIAA Journal*, 30:311–324, 1992.

[118] D.C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries Inc., 1993.

[119] M.W. Johnson and A.H. Ercan. A physical model for bypass transition. *International Journal of Heat and Fluid Flow*, 20:95–104, 1999.

[120] K. Suluksna and E. Juntasaro. Assessment of intermittency transport equations for modeling transition in boundary layers subjected to freestream turbulence. *International Journal of Heat and Fluid Flow*, 29:48–61, 2008.

[121] M. Klein, A. Sadiki, and J. Janicka. A digital filter based generation of inflow data for spatially developing direct numerical or large eddy simulations. *Journal of Computational Physics*, 186(2):652–665, 2003.

[122] T.S. Lund, X. Wu, and K.D. Squires. Generation of turbulent inflow data for spatially-developing boundary layer simulations. *Journal of Computational Physics*, 140(2):233–258, 1998.

[123] N. Jarrin, S. Benhamadouche, Y. Addad, and D. Laurence. Synthetic turbulent inflow conditions for large eddy simulation. In *Proceedings of the 4th International Turbulence, Heat and Mass Transfer Conference, Antalya, Turkey*, 2003.

[124] G.R. Tabor and M.H. Baba-Ahmadi. Inlet conditions for large eddy simulation: A review. *Computers & Fluids*, 39(4):553–567, 2010.

[125] OpenCFD. *OpenFOAM - The Open Source CFD Toolbox - User Guide*. OpenCFD Ltd., United Kingdom, 2.4.0 edition, 2015.

[126] A. Travin, M. Shur, M. Strelets, and P. Spalart. Detached-eddy simulations past a circular cylinder. *Flow, Turbulence and Combustion*, 63(1-4):293–313, 2000.

[127] M. Shur, P.R. Spalart, M. Strelets, and A. Travin. Detached-eddy simulation of an airfoil at high angle of attack. *Engineering turbulence modelling and experiments*, 4:669–678, 1999.

[128] W. Haase, B. Aupoix, U. Bunge, and D. Schwamborn. *FLOMANIA - A European Initiative on Flow Physics Modelling: Results of the European-Union funded project, 2002 - 2004*. Notes on Numerical Fluid Mechanics and Multidisciplinary Design. Springer Berlin Heidelberg, 2006.

[129] A. Travin, M. Shur, M. Strelets, and P.R. Spalart. Physical and numerical upgrades in the detached-eddy simulation of complex turbulent flows. In R. Friedrich and W. Rodi, editors, *Advances in LES of Complex Flows*, volume 65 of *Fluid Mechanics and Its Applications*, pages 239–254. Springer Netherlands, 2002.

[130] J.R. Edwards and S. Chandra. Comparison of eddy viscosity-transport turbulence models for three-dimensional, shock-separated flowfields. *AIAA journal*, 34(4):756–763, 1996.

[131] T. Rung, U. Bunge, M. Schatz, and F. Thiele. Restatement of the Spalart-Allmaras eddy-viscosity model in strain-adaptive formulation. *AIAA journal*, 41(7):1396–1399, 2003.

[132] M. Shur, P.R. Spalart, M. Strelets, and A. Travin. Modification of SA subgrid model in DES aimed to prevent activation of the low-Re terms in LES mode. *Proceedings of workshop on DES, St.Petersburg*, pages 2–3, 2003.

[133] C. Mockett, B. Greschner, T. Knacke, R. Perrin, J. Yan, and F. Thiele. Demonstration of improved des methods for generic and industrial applications. In Shia-Hui Peng and Werner Haase, editors, *Advances in Hybrid RANS-LES Modelling*, volume 97 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 222–231. Springer Berlin Heidelberg, 2008.

[134] J. Yan, C. Mockett, and F. Thiele. Investigation of alternative length scale substitutions in detached-eddy simulation. *Flow, Turbulence and Combustion*, 74(1):85–102, 2005.

[135] J.C. Kok, H.S. Dol, B. Oskam, and H. van der Ven. Extra-large eddy simulation of massively separated flows. *AIAA paper*, 264, 2004.

[136] M. Braza, R. Perrin, and Y. Hoarau. Turbulence properties in the cylinder wake at high Reynolds numbers. *Journal of Fluids and Structures*, 22(6-7):757–771, 2006.

[137] F. El Akoury. Analyse physique des effets de rotation de paroi en écoulements transitionnels et modélisation d'écoulements turbulents autour de structures portantes. Technical Report PhD Thesis, Institut de Mécanique des Fluides de Toulouse - IMFT (Toulouse, France), 2007.

[138] S.H. Peng. Towards detached eddy simulation modelling using a k-equation turbulence model. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5-8, 2006*. Delft University of Technology; European Community on Computational Methods in Applied Sciences (ECCOMAS), 2006.

[139] H.M. Lubcke, Th. Rung, and F. Thiele. Prediction of the spreading mechanism of 3D turbulent wall jets with explicit Reynolds-stress closures. *International Journal of Heat and Fluid Flow*, 24(4):434–443, 2003.

[140] F.S. Lien and M.A. Leschziner. Computational modelling of 3D turbulent flow in s-diffuser and transition ducts. In F. Martelli, W. Rodi, editor, *Engineering Turbulence Modelling and Experiments*, pages 217 – 228. Elsevier, Oxford, 1993.

[141] D.C. Wilcox. Re-assessment of the scale-determining equation for advanced turbulence models. *AIAA Journal*, 26(11):1299–1310, 1988.

[142] F.R. Menter and M. Kuntz. Adaptation of eddy-viscosity turbulence models to unsteady separated flow behind vehicles. In Rose McCallen, Fred Browand, and James Ross, editors, *The Aerodynamics of Heavy Vehicles: Trucks, Buses, and Trains*, volume 19 of *Lecture Notes in Applied and Computational Mechanics*, pages 339–352. Springer Berlin Heidelberg, 2004.

[143] H. Medina, A. Beechook, J. Saul, S. Porter, S. Aleksandrova, and S. Benjamin. Open source Computational Fluid Dynamics using OpenFOAM. In *Proceedings of the Royal Aeronautical Society General Aviation Conference on Light Aircraft Design Methods and Tools 2015, London, UK, November 16, 2015*, 2015.

[144] B.E. Launder and B.I. Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in Heat and Mass Transfer*, 1(2):131–138, 1974.

[145] V. C. Patel, W. Rodi, and G. Scheuerer. Turbulence models for near-wall and low-Reynolds number flows - A review. *AIAA journal*, 23(9):1308–1319, 1985.

[146] M.A. Cotton and J.D. Jackson. Vertical tube air flows in the turbulent mixed convection regime calculated using a low-Reynolds-number $k-\epsilon$ model. *International Journal of Heat and Mass Transfer*, 33(2):275–286, 1990.

[147] S.S. Thakre and J.B. Joshi. CFD modeling of heat transfer in turbulent pipe flows. *AIChE Journal*, 46(9):1798–1812, 2000.

[148] W.S. Kim, S. He, and J.D. Jackson. Assessment by comparison with DNS data of turbulence models used in simulations of mixed convection. *International Journal of Heat and Mass Transfer*, 51(5-6):1293–1312, 2008.

[149] P.A. Durbin. Near-wall turbulence closure modeling without damping functions. *Theoretical and Computational Fluid Dynamics*, 3:1–13, 1991.

[150] F.S. Lien and G. Kalitzin. Computations of transonic flow with the v^2-f turbulence model. *International Journal of Heat and Fluid Flow*, 22:53–61, 2001.

[151] L. Davidson, P. Nielsen, and A. Sveningsson. Modifications of the v^2-f model for computing the flow in a 3D wall jet. *Turbulence, Heat and Mass Transfer*, 4:577–584, 2003.

[152] M.M. Gibson and A.A. Dafa'Alla. Two-equation model for turbulent wall flow. *AIAA Journal*, 33(8):1514–1518, 1995.

Appendix A

RANS Models

A.1 $k - \epsilon$ model

The standard $k - \epsilon$ model is known to be one of most common and widely used two-equation turbulence models, for simulation of mean flow behaviour and characteristics for turbulent flows. This model utilises two transport equations, the turbulent kinetic energy, k and the turbulence specific dissipation rate, ϵ.

The turbulent kinetic energy, k equation reads

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = \tau_{ij} \frac{\partial U_i}{\partial U_j} - \epsilon + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_T}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] \tag{A.1}$$

The turbulence specific dissipation rate ϵ equation reads

$$\frac{\partial \epsilon}{\partial t} + U_j \frac{\partial \epsilon}{\partial x_j} = C_{\epsilon 1} \frac{\epsilon}{k} \tau_{ij} \frac{\partial U_i}{\partial U_j} - C_{\epsilon 2} \frac{\epsilon^2}{k} + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_T}{\sigma_\epsilon} \right) \frac{\partial \epsilon}{\partial x_j} \right] \tag{A.2}$$

The turbulent eddy viscosity is defined by

$$\nu_T = C_\mu \frac{k^2}{\epsilon} \tag{A.3}$$

The model empirical constants defined by Jones and Launder [27] are given by

$$C_\mu = 0.09 \qquad C_{\epsilon 1} = 1.44 \qquad C_{\epsilon 2} = 1.92 \qquad \sigma_k = 1.0 \qquad \sigma_\epsilon = 1.3 \qquad \alpha_K = 0.9$$

A.2 $k - \omega$ model

The $k - \omega$ turbulence model of Wilcox [141] is a two-equation model eddy-viscosity model.

The turbulent kinetic energy, k equation reads

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = \tau_{ij} \frac{\partial U_i}{\partial U_j} - \beta^* k\omega + \frac{\partial}{\partial x_j} \left[(\nu + \sigma_k \nu_T) \frac{\partial k}{\partial x_j} \right] \tag{A.4}$$

The turbulence specific dissipation rate ω equation reads

$$\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha_1 \frac{\omega}{k} \tau_{ij} \frac{\partial U_i}{\partial U_j} - \beta \omega^2 + \frac{\partial}{\partial x_j} \left[(\nu + \sigma_\omega \nu_T) \frac{\partial \omega}{\partial x_j} \right] \tag{A.5}$$

The turbulent eddy viscosity is defined by

$$\nu_T = \frac{k}{\omega} \tag{A.6}$$

The model empirical constants defined by Wilcox [141] are given by

$$\sigma_k = 0.5 \qquad \sigma_\omega = 0.5 \qquad \beta = \frac{3}{40} \qquad \beta^* = 0.09 \qquad \alpha_1 = \frac{5}{9}$$

It is to be noted that α_1 coefficient is usually referred to as γ in a wide range of references. The term γ is usually employed in the attempt to achieve a reasonable value for the Von Kármán constant ($\kappa = 0.41$), from the following expression

$$\gamma = \frac{\beta}{\beta^*} - \frac{\sigma_\omega \kappa^2}{\sqrt{\beta^*}} \tag{A.7}$$

A.3 $k-\omega$ SST model

The Shear Stress Transport (SST) model combines several elements of existing two-equation models. It is a two-layer model which is based upon Wilcox's $k-\omega$ model in the inner region of boundary layers and switches to $k-\epsilon$ model in the outer region of boundary layers and in mixing regions. The outer $k-\epsilon$ model is transformed to provide a second set of $k-\omega$ equations with a blending function used to transition between the two sets of equations.

The model equations read

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = \tau_{ij} \frac{\partial U_i}{\partial U_j} - \beta^* k\omega + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_k \nu_T)\frac{\partial k}{\partial x_j}\right] \tag{A.8}$$

and

$$\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha_1 \frac{\omega}{k}\nu_T S^2 - \beta\omega^2 + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_\omega \nu_T)\frac{\partial \omega}{\partial x_j}\right] + 2(1 - F_1)\frac{\rho\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \tag{A.9}$$

where F_1 is the blending function and is defined by

$$F_1 = \tanh\left\{\left\{\min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right), \frac{4\sigma_{\omega 2}k}{CD_{k\omega}y^2}\right]\right\}^4\right\} \tag{A.10}$$

and

$$CD_{k\omega} = \max\left(2\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, 10^{-10}\right) \tag{A.11}$$

Away from the wall, $F_1 = 0$, the SST model switches to $k-\epsilon$ model and in the boundary layer, $F_1 = 1$, the model switches to $k-\omega$ model.

The turbulent eddy viscosity is defined as

$$\nu_t = \frac{\alpha_1 k}{\max(\alpha_1\omega, \Omega F_2)} \tag{A.12}$$

where

$$F_2 = \tanh\left[\left[\max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right)\right]^2\right] \tag{A.13}$$

The model empirical constants defined by Menter [38] are

$$\sigma_{k1} = 0.85 \qquad \sigma_{\omega 1} = 0.5 \qquad \beta_1 = \frac{3}{40} \qquad \sigma_{k2} = 1.0$$
$$\sigma_{\omega 2} = 0.856 \qquad \beta_2 = 0.0828 \qquad \beta^* = 0.09 \qquad \alpha_1 = \frac{5}{9}$$

A.4 Launder-Sharma $k-\epsilon$ model

The Launder-Sharma $k-\epsilon$ turbulence model [144] is classified as one most widely used low Reynolds number model for near-wall modelling application. It is a modified version of the standard $k-\epsilon$ model of Jones and Launder [27]. The performance of the Launder-Sharma $k-\epsilon$ model has been shown to be superior to other Low Reynolds number models when compared to experimental data [145, 146] as well as DNS data [147, 148]. The model has two transport equations, which are based on the equations of the standard $k-\epsilon$ model. The model is described as follows

The turbulent kinetic energy, k equation is given by

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = P - \epsilon - D + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \tag{A.14}$$

The turbulent energy dissipation, ϵ equation can be written as

$$\frac{\partial \tilde{\epsilon}}{\partial t} + U_j \frac{\partial \tilde{\epsilon}}{\partial x_j} = C_{\epsilon 1} f_1 P \frac{\tilde{\epsilon}}{k} - C_{\epsilon 2} f_2 \frac{\tilde{\epsilon}^2}{k} + E + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\frac{\partial \tilde{\epsilon}}{\partial x_j}\right] \tag{A.15}$$

where $\tilde{\epsilon}$ is a modified turbulence dissipation rate ϵ. There is almost no difference between $\tilde{\epsilon}$ and ϵ as their values are effectively the same for most regimes of the flow. They only differ in the near-wall region.

$$\tilde{\epsilon} = (\epsilon - D) \tag{A.16}$$

and

$$P = \tau_{ij} \frac{\partial u_i}{\partial x_j} \tag{A.17}$$

The turbulent eddy viscosity reads

$$\nu_T = C_\mu f_\mu \frac{k^2}{\epsilon} \tag{A.18}$$

The damping functions within the ϵ equation are defined as follows

$$f_\mu = \exp\frac{-3.4}{\left(1 + Re_t/50\right)^2} \qquad f_1 = 1 \qquad f_2 = 1 - 0.3\exp\left(-Re_t^2\right)$$

where

$$Re_t = \frac{k^2}{\nu\tilde{\epsilon}}$$

The source terms D and E are defined as

$$D = 2\nu\left(\frac{\partial\sqrt{k}}{\partial y}\right)^2 \qquad E = 2\nu\nu_t\left(\frac{\partial^2 u}{\partial y^2}\right)^2$$

The model empirical constants defined by Launder and Sharma [144] are

$$C_{\epsilon_1} = 1.44 \qquad C_{\epsilon_2} = 1.92 \qquad C_\mu = 0.09 \qquad \sigma_k = 1.0 \qquad \sigma_\epsilon = 1.3$$

A.5 $v^2 - f$ model

The $v^2 - f$ turbulence model, originally proposed by Durbin [149] is based on the standard $k - \epsilon$ model and consists of four transport equations: The turbulent kinetic energy k and the specific dissipation rate ϵ equations are simply the transport equations of the standard $k - \epsilon$ model, i.e. equations (3.28) and (3.29). The other two additional equations are for the imaginary turbulent normal stress v^2 and the elliptical relaxation function f. These two additional equations accounts for the additional feature of the $v^2 - f$ model, that incorporates near-wall anisotropy of turbulence as well as non-local pressure-strain effects. Rather than using the turbulent kinetic energy k, the $v^2 - f$ model utilises a velocity scale v^2 for the evaluation of the eddy viscosity term. v^2 is usually referred to as the velocity fluctuations normal to the streamlines.

The model described below is a modified version of the original $v^2 - f$ model by Lien and Kalitzin [150], allowing a segregated solution for the transport equations. This model was further modified by Davidson et al. [151] to limit the imaginary velocity scale to $2/3k$. In addition to the equations of the standard $k - \epsilon$ turbulence model, the turbulent quantities are computed from two other equations.

They are the transport equation for v^2 reads

$$\frac{\partial \overline{U_j v'^2}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\frac{\partial \overline{v'^2}}{\partial x_j}\right] + kf - 6\frac{\overline{v^2}}{k}\epsilon \tag{A.19}$$

The equation for the elliptical relaxation function, f reads

$$L^2\frac{\partial^2 f}{\partial x_j \partial x_j} - f = \frac{1}{T}\left[(C_1 - 6)\frac{\overline{v'^2}}{k} - \frac{2}{3}(C_1 - 1)\right] - C_2\frac{P_k}{k} \tag{A.20}$$

where

$$P_k = \nu_t\left(\frac{\partial \overline{U_i}}{\partial x_j} + \frac{\partial \overline{U_j}}{\partial x_i}\right)\frac{\partial \overline{U_i}}{\partial x_j} \tag{A.21}$$

The turbulence length scale L is given by

$$L = C_L \max\left[\frac{k^{3/2}}{\epsilon}, C_\eta\left(\frac{\nu^3}{\epsilon}\right)^{1/4}\right] \tag{A.22}$$

The turbulence time scale T is given by

$$T = \max\left[\frac{k}{\epsilon}, 6\sqrt{\frac{\nu}{\epsilon}}\right] \tag{A.23}$$

The turbulent eddy viscosity is given by

$$\nu_t = C_\mu \overline{v^2} T \tag{A.24}$$

The standard k and ϵ equations are also solved without the use of damping functions. The boundary conditions at the walls are

$$k = \overline{v^2} = f = 0, \epsilon = 2\nu k/y^2$$

The empirical constants are

$C_\mu = 0.22$ $C_{\epsilon 2} = 1.9$ $\sigma_k = 1$ $\sigma_\epsilon = 1.3$ $C_1 = 1.4$ $C_2 = 0.3$ $C_L = 0.23$ $C_\eta = 70$

A.6 $q-\zeta$ model

The $q-\zeta$ turbulence model is a modified variant of the standard $k-\epsilon$ model, developed to eliminate the dependence of k and ϵ on y^2 near the wall [152]. This model was introduced to solve the problem of standard $k-\epsilon$ model where neither the k equation nor the ϵ equation are well adapted to integration through the viscous layers to the wall [152]. This model is classified in the low Reynolds number model category and it utilises damping functions based on the approaches of Jones and Launder [27] and Launder and Sharma [144].

The eddy viscosity term of the $q-\zeta$ model is defined as

$$\nu_T = C_\mu f_\mu \left(\frac{k^2}{\epsilon}\right) \tag{A.25}$$

The new variables introduced in the $k-\epsilon$ model are q and ζ

$$q \equiv \sqrt{k} \tag{A.26}$$

The rate of destruction is given by

$$\zeta \equiv \frac{\tilde{\epsilon}}{2q} \tag{A.27}$$

where

$$\tilde{\epsilon} = \epsilon - 2\nu\frac{\partial q^2}{\partial y} \tag{A.28}$$

Based on the new variables, the turbulent eddy viscosity can be written as

$$\nu_T = C_\mu f_\mu \left(\frac{q^3}{2\zeta}\right) \tag{A.29}$$

The equation of q is given by

$$U_i\frac{\partial q}{\partial x_i} = \frac{\partial}{\partial x_j}\left[\left(\frac{\nu_{T_{q-\zeta}}}{\sigma_q}+\nu\right)\frac{\partial q}{\partial x_j}\right] + \frac{P}{2q} - \frac{\tilde{\epsilon}}{2q} \tag{A.30}$$

The equation of ζ can be written as

$$U_i\frac{\partial \zeta}{\partial x_i} = \frac{\partial}{\partial x_j}\left[\left(\frac{\nu_{T_{q-\zeta}}}{\sigma_\zeta}+\nu\right)\frac{\partial \zeta}{\partial x_j}\right] + \left(C_{\epsilon 1}f_{\epsilon 1} - \frac{1}{2}\right)\frac{P\zeta}{q^2} - \left(C_{\epsilon 2}f_{\epsilon 2} - \frac{1}{2}\right)\frac{2\zeta^2}{q} + \psi' + \xi' \tag{A.31}$$

where

$\psi' = \psi/2q$ and ξ' are optional viscous terms.

The model constants [152] are defined below

$$C_\mu = 0.09 \qquad C_{\epsilon 1} = 1.35 \qquad C_{\epsilon 2} = 1.80 \qquad \sigma_k = 1.0 \qquad \sigma_q = 1.0 \qquad \sigma_\zeta = 1.3$$

Appendix B

The SIMPLEC algorithm OpenFOAM implementation

B.1 simplecFoam.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2015 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    simplecFoam

Description
    Steady-state solver for incompressible, turbulent flow

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "singlePhaseTransportModel.H"
#include "RASModel.H"
#include "simpleControl.H"
#include "fvIOoptionList.H"
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    #include "createFields.H"
    #include "createFvOptions.H"
    #include "initContinuityErrs.H"

    simpleControl simple(mesh);

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nStarting time loop\n" << endl;

    while (simple.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        // --- Pressure-velocity SIMPLE corrector
        {
            #include "UEqn.H"
            #include "pEqn.H"
        }

        turbulence->correct();

        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return 0;
}


// ************************************************************************* //
```

B.2 pEqn.H

```
{
    volScalarField rAU(1.0/UEqn().A());
    volScalarField rAtU(1.0/(1.0/rAU - UEqn().H1()));

    volVectorField HbyA("HbyA", U);
    HbyA = rAU*UEqn().H();
    UEqn.clear();

    surfaceScalarField phiHbyA("phiHbyA", fvc::interpolate(HbyA) & mesh.Sf());
    //surfaceScalarField phid("phid", fvc::interpolate(HbyA) & mesh.Sf());

    fvOptions.makeRelative(phiHbyA);

    adjustPhi(phiHbyA, U, p);

    phiHbyA += fvc::interpolate((rAtU - rAU))*fvc::snGrad(p)*mesh.magSf();
    HbyA -= (rAU - rAtU)*fvc::grad(p);

    // Non-orthogonal pressure corrector loop
    while (simple.correctNonOrthogonal())
    {
        fvScalarMatrix pEqn
        (
            //fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
  fvm::laplacian(rAtU, p) == fvc::div(phiHbyA)
        );

        pEqn.setReference(pRefCell, pRefValue);

        pEqn.solve();

        if (simple.finalNonOrthogonalIter())
        {
            phi = phiHbyA - pEqn.flux();
        }
    }

    #include "continuityErrs.H"

    // Explicitly relax pressure for momentum corrector
    p.relax();

    // Momentum corrector
    U = HbyA - rAtU*fvc::grad(p);
    U.correctBoundaryConditions();
    fvOptions.correct(U);
}
```

B.3 UEqn.H

```
// Momentum predictor

tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U)
  + turbulence->divDevReff(U)
  ==
    fvOptions(U)
);

UEqn().relax();

fvOptions.constrain(UEqn());

solve(UEqn() == -fvc::grad(p));

fvOptions.correct(U);
```

B.4 createFields.H

```
Info<< "Reading field p\n" << endl;
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field U\n" << endl;
volVectorField U
(
    IOobject
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

#include "createPhi.H"


label pRefCell = 0;
scalar pRefValue = 0.0;
setRefCell(p, mesh.solutionDict().subDict("SIMPLE"), pRefCell, pRefValue);

singlePhaseTransportModel laminarTransport(U, phi);

autoPtr<incompressible::RASModel> turbulence
(
    incompressible::RASModel::New(U, phi, laminarTransport)
);
```

Appendix C

The $k_T - k_L - \omega$ DES model OpenFOAM implementation

C.1 kklOmegaDES.H

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2015 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Class
    Foam::incompressible::LESModels::kkLOmegaDES

Group
    grpIcoLESTurbulence

Description
    k-kl-Omega-DES turbulence model for incompressible flows.

        The baseline RAS turbulence model used is described in:

    \verbatim
```

```
        Walters, D. K., & Cokljat, D. (2008).
        A three-equation eddy-viscosity model for Reynolds-averaged
        Navier-Stokes simulations of transitional flow.
        Journal of Fluids Engineering, 130(12), 121401.
    \endverbatim

    However the paper contains several errors which must be corrected for the
    RAS model to operate correctly as explained in

       \verbatim
        Medina, H., & Early, J. (2014).
        Modelling transition due to backward-facing steps using the laminar
        kinetic energy concept.
        European Journal of Mechanics B/Fluids, 44(2014), 60-68.
    \endverbatim
-------------------------------------------------------------------------------
DISCLAIMER:
    This model is part of the requirements for the degree of Master by Research
    in Aerospace Egineering at Coventry University, under the supervision of
    Dr. Humberto Medina, Senior Lecturer in Aerospace Engineering.
-------------------------------------------------------------------------------

SourceFiles
    kkLOmegaDES.C

\*---------------------------------------------------------------------------*/

#ifndef kkLOmegaDES_H
#define kkLOmegaDES_H

#include "LESModel.H"
#include "volFields.H"
#include "wallDist.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
namespace incompressible
{
namespace LESModels
{

/*---------------------------------------------------------------------------*\
                        Class kkLOmegaDES Declaration
\*---------------------------------------------------------------------------*/

class kkLOmegaDES
:
    public LESModel
{
    // Private Member Functions

        //- Update sub-grid scale fields
        void updateSubGridScaleFields(const volScalarField& nuts, const volScalarField& nutl);

        // Disallow default bitwise copy construct and assignment
        kkLOmegaDES(const kkLOmegaDES&);
        kkLOmegaDES& operator=(const kkLOmegaDES&);
```

```
protected:

    // Protected Member Functions

// kklOmega Private member functions

        tmp<volScalarField> fv(const volScalarField& Ret) const;

        tmp<volScalarField> fINT() const;

        tmp<volScalarField> fSS(const volScalarField& omega) const;

        tmp<volScalarField> Cmu(const volScalarField& S) const;

        tmp<volScalarField> BetaTS(const volScalarField& ReOmega) const;

        tmp<volScalarField> fTaul
        (
            const volScalarField& lambdaEff,
            const volScalarField& ktL,
            const volScalarField& omega
        ) const;

        tmp<volScalarField> alphaT
        (
            const volScalarField& lambdaEff,
            const volScalarField& fv,
            const volScalarField& ktS
        ) const;

        tmp<volScalarField> fOmega
        (
            const volScalarField& lambdaEff,
            const volScalarField& lambdaT
        ) const;

        tmp<volScalarField> phiBP(const volScalarField& omega) const;

        tmp<volScalarField> phiNAT
        (
            const volScalarField& ReOmega,
            const volScalarField& fNatCrit
        ) const;

        tmp<volScalarField> FDES() const;

        // Model constants

            dimensionedScalar A0_;
            dimensionedScalar As_;
            dimensionedScalar Av_;
            dimensionedScalar Abp_;
            dimensionedScalar Anat_;
            dimensionedScalar Ats_;
            dimensionedScalar CbpCrit_;
            dimensionedScalar Cnc_;
            dimensionedScalar CnatCrit_;
```

```
            dimensionedScalar Cint_;
            dimensionedScalar CtsCrit_;
            dimensionedScalar CrNat_;
            dimensionedScalar C11_;
            dimensionedScalar C12_;
            dimensionedScalar CR_;
            dimensionedScalar CalphaTheta_;
            dimensionedScalar Css_;
            dimensionedScalar CtauL_;
            dimensionedScalar Cw1_;
            dimensionedScalar Cw2_;
            dimensionedScalar Cw3_;
            dimensionedScalar CwR_;
            dimensionedScalar Clambda_;
            dimensionedScalar CmuStd_;
            dimensionedScalar Prtheta_;
            dimensionedScalar Sigmak_;
            dimensionedScalar Sigmaw_;
    dimensionedScalar CDES_;
    dimensionedScalar kMin_;
    dimensionedScalar omegaMin_;

        // Fields

            volScalarField kt_;
            volScalarField kl_;
            volScalarField omega_;
            volScalarField nuSgs_;
            volScalarField FDESout_;


            //- Wall distance
            wallDist y_;


public:

    //- Runtime type information
    TypeName("kkLOmegaDES");


    // Constructors

        //- Construct from components
        kkLOmegaDES
        (
            const volVectorField& U,
            const surfaceScalarField& phi,
            transportModel& transport,
            const word& turbulenceModelName = turbulenceModel::typeName,
            const word& modelName = typeName
        );


    //- Destructor
    virtual ~kkLOmegaDES()
    {}
```

```
// Member Functions

    //- Return the effective diffusivity for k
    tmp<volScalarField> DkEff(const volScalarField& alphaT) const
    {
        return tmp<volScalarField>
        (
            new volScalarField("DkEff", alphaT/Sigmak_ + nu())
        );
    }

    //- Return the effective diffusivity for omega
    tmp<volScalarField> DomegaEff(const volScalarField& alphaT) const
    {
        return tmp<volScalarField>
        (
            new volScalarField("DomegaEff", alphaT/Sigmaw_ + nu())
        );
    }

    //- Return SGS kinetic energy
    virtual tmp<volScalarField> k() const
    {
        return (kt_ + kl_); // this needs changing!!!!!
    }

    //- Return the laminar kinetic energy
    virtual tmp<volScalarField> kl() const
    {
        return kl_;
    }

    //- Return the turbulence kinetic energy
    virtual tmp<volScalarField> kt() const
    {
        return kt_;
    }

    //- Return the turbulence specific dissipation rate
    virtual tmp<volScalarField> omega() const
    {
        return omega_;
    }


    //- Return sub-grid disipation rate
    virtual tmp<volScalarField> epsilon() const;

    //- Return SGS viscosity
    virtual tmp<volScalarField> nuSgs() const
    {
        return nuSgs_;
    }

    virtual tmp<volScalarField> FDESout() const
    {
        return FDESout_;
    }
```

```
        //- Return the sub-grid stress tensor.
        virtual tmp<volSymmTensorField> B() const;

        //- Return the effective sub-grid turbulence stress tensor
        //  including the laminar stress
        virtual tmp<volSymmTensorField> devReff() const;

        //- Return the deviatoric part of the effective sub-grid
        //  turbulence stress tensor including the laminar stress
        virtual tmp<fvVectorMatrix> divDevReff(volVectorField& U) const;

        //- Return the deviatoric part of the effective sub-grid
        //  turbulence stress tensor including the laminar stress
        virtual tmp<fvVectorMatrix> divDevRhoReff
        (
            const volScalarField& rho,
            volVectorField& U
        ) const;

        //- Solve the turbulence equations (k-w) and correct the turbulence
        //  viscosity
        virtual void correct(const tmp<volTensorField>& gradU);

        //- Read LESProperties dictionary
        virtual bool read();
};


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace LESModels
} // End namespace incompressible
} // End namespace Foam

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //
```

C.2 kklOmegaDES.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2015 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------------*/

#include "kkLOmegaDES.H"
#include "addToRunTimeSelectionTable.H"
#include "wallDist.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
namespace incompressible
{
namespace LESModels
{

// * * * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * * * //

defineTypeNameAndDebug(kkLOmegaDES, 0);
addToRunTimeSelectionTable(LESModel, kkLOmegaDES, dictionary);

// * * * * * * * * * * * * * * * Protected Member Functions  * * * * * * * * * * * * //

void kkLOmegaDES::updateSubGridScaleFields(const volScalarField& nuts, const volScalarField& nutl)
{
    nuSgs_ = nuts + nutl;
    nuSgs_.correctBoundaryConditions();
}


/// kklOmega Fuctions

tmp<volScalarField> kkLOmegaDES::fv(const volScalarField& Ret) const
{
    return(1.0 - exp(-sqrt(Ret)/Av_));
```

```
}


tmp<volScalarField> kkLOmegaDES::fINT() const
{
    return
    (
        min
        (
            kt_/(Cint_*(kl_ + kt_ + kMin_)),
            dimensionedScalar("1.0", dimless, 1.0)
        )
    );
}


tmp<volScalarField> kkLOmegaDES::fSS(const volScalarField& Omega) const
{
    return(exp(-sqr(Css_*nu()*Omega/(kt_ + kMin_))));
}


tmp<volScalarField> kkLOmegaDES::Cmu(const volScalarField& S) const
{
    return(1.0/(A0_ + As_*(S/(omega_ + omegaMin_))));
}


tmp<volScalarField> kkLOmegaDES::BetaTS(const volScalarField& ReOmega) const
{
    return(scalar(1) - exp(-sqr(max(ReOmega - CtsCrit_, scalar(0)))/Ats_));
}


tmp<volScalarField> kkLOmegaDES::fTaul
(
    const volScalarField& lambdaEff,
    const volScalarField& ktL,
    const volScalarField& Omega
) const
{
    return
    (
        scalar(1)
      - exp
        (
            -CtauL_*ktL
          /
            (
                sqr
                (
                    lambdaEff*Omega
                  + dimensionedScalar
                    (
                        "ROOTVSMALL",
                        dimLength*inv(dimTime),
                        ROOTVSMALL
                    )
                )
```

```
            )
        )
    );
}


tmp<volScalarField> kkLOmegaDES::alphaT
(
    const volScalarField& lambdaEff,
    const volScalarField& fv,
    const volScalarField& ktS
) const
{
    return(fv*CmuStd_*sqrt(ktS)*lambdaEff);
}


tmp<volScalarField> kkLOmegaDES::fOmega
(
    const volScalarField& lambdaEff,
    const volScalarField& lambdaT
) const
{
    return
    (
        scalar(1)
      - exp
        (
           -0.41
           *pow4
            (
                lambdaEff
              / (
                    lambdaT
                  + dimensionedScalar
                    (
                        "ROTVSMALL",
                        lambdaT.dimensions(),
                        ROOTVSMALL
                    )
                )
            )
        )
    );
}


tmp<volScalarField> kkLOmegaDES::phiBP(const volScalarField& Omega) const
{
    return
    (
        min
        (
            max
            (
                kt_/nu()
             / (
                    Omega
                  + dimensionedScalar
```

```
                    (
                        "ROTVSMALL",
                        Omega.dimensions(),
                        ROOTVSMALL
                    )
                )
              - CbpCrit_,
                scalar(0)
            ),
            scalar(50.0)
        )
    );
}


tmp<volScalarField> kkLOmegaDES::phiNAT
(
    const volScalarField& ReOmega,
    const volScalarField& fNatCrit
) const
{
    return
    (
        max
        (
            ReOmega
          - CnatCrit_
          / (
                fNatCrit + dimensionedScalar("ROTVSMALL", dimless, ROOTVSMALL)
            ),
            scalar(0)
        )
    );
}

// F_DES term definition

tmp<volScalarField> kkLOmegaDES::FDES() const
{
    return max(scalar(1), sqrt(kt_)/(CDES_*omega_*delta()));
}

// * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * * //

kkLOmegaDES::kkLOmegaDES
(
    const volVectorField& U,
    const surfaceScalarField& phi,
    transportModel& transport,
    const word& turbulenceModelName,
    const word& modelName
)
:
    LESModel(modelName, U, phi, transport, turbulenceModelName),

    A0_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
```

```
            "A0",
            coeffDict_,
            4.04
        )
    ),
    As_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "As",
            coeffDict_,
            2.12
        )
    ),
    Av_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Av",
            coeffDict_,
            6.75
        )
    ),
    Abp_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Abp",
            coeffDict_,
            0.6
        )
    ),
    Anat_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Anat",
            coeffDict_,
            200
        )
    ),
    Ats_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Ats",
            coeffDict_,
            200
        )
    ),
    CbpCrit_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CbpCrit",
            coeffDict_,
            1.2
        )
    ),
```

```
    Cnc_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cnc",
            coeffDict_,
            0.1
        )
    ),
    CnatCrit_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CnatCrit",
            coeffDict_,
            1250
        )
    ),
    Cint_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cint",
            coeffDict_,
            0.75
        )
    ),
    CtsCrit_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CtsCrit",
            coeffDict_,
            1000
        )
    ),
    CrNat_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CrNat",
            coeffDict_,
            0.02
        )
    ),
    C11_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "C11",
            coeffDict_,
            3.4e-6
        )
    ),
    C12_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "C12",
```

```
            coeffDict_,
            1.0e-10
        )
    ),
    CR_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CR",
            coeffDict_,
            0.12
        )
    ),
    CalphaTheta_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CalphaTheta",
            coeffDict_,
            0.035
        )
    ),
    Css_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Css",
            coeffDict_,
            1.5
        )
    ),
    CtauL_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CtauL",
            coeffDict_,
            4360
        )
    ),
    Cw1_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cw1",
            coeffDict_,
            0.44
        )
    ),
    Cw2_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cw2",
            coeffDict_,
            0.92
        )
    ),
    Cw3_
```

```
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "Cw3",
        coeffDict_,
        0.3
    )
),
CwR_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "CwR",
        coeffDict_,
        1.5
    )
),
Clambda_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "Clambda",
        coeffDict_,
        2.495
    )
),
CmuStd_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "CmuStd",
        coeffDict_,
        0.09
    )
),
Prtheta_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "Prtheta",
        coeffDict_,
        0.85
    )
),
Sigmak_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "Sigmak",
        coeffDict_,
        1
    )
),
Sigmaw_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "Sigmaw",
        coeffDict_,
```

```
            1.17
        )
    ),

    CDES_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CDES",
            coeffDict_,
            0.61
        )
    ),

    kMin_("kMin", sqr(dimVelocity), SMALL),

    omegaMin_("omegaMin", dimless/dimTime, SMALL),

    kt_
    (
        IOobject
        (
            "kt",
            runTime_.timeName(),
            mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_
    ),

    kl_
    (
        IOobject
        (
            "kl",
            runTime_.timeName(),
            mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_
    ),

    omega_
    (
        IOobject
        (
            "omega",
            runTime_.timeName(),
            mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_
    ),

    nuSgs_
    (
```

```
        IOobject
        (
            "nuSgs",
            runTime_.timeName(),
            mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_
    ),

    FDESout_
    (
        IOobject
        (
            "FDES",
            runTime_.timeName(),
            mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_,
        dimensionedScalar("FDES", dimless, 0.0)
    ),

    y_(mesh_)


{
    kMin_.readIfPresent(*this);
    omegaMin_.readIfPresent(*this);

    bound(kt_, kMin_);
    bound(kl_, kMin_);
    bound(omega_, omegaMin_);

    updateSubGridScaleFields(kt_/max(omega_, omegaMin_), kl_/max(omega_, omegaMin_));

    printCoeffs();
}


// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //

void kkLOmegaDES::correct(const tmp<volTensorField>& gradU)
{
    LESModel::correct(gradU);

    if (mesh_.changing())
    {
        y_.correct();
    }

    volScalarField S2(2.0*magSqr(symm(gradU())));
    volScalarField Omega(sqrt(2.0)*mag(skew(gradU())));
    gradU.clear();
```

```
/// kklOmega functions and definitions

     volScalarField lambdaT(sqrt(kt_)/(omega_ + omegaMin_));

     volScalarField lambdaEff(min(Clambda_*y_, lambdaT));

     volScalarField fw
    (
        pow
        (
            lambdaEff
           /(lambdaT + dimensionedScalar("SMALL", dimLength, ROOTVSMALL)),
            2.0/3.0
        )
    );

     volScalarField ktS(fSS(Omega)*fw*kt_);

     volScalarField nuts // Small scale eddy viscosity
    (
        fv(sqr(fw)*kt_/nu()/(omega_ + omegaMin_))
       *fINT()
       *Cmu(sqrt(S2))*sqrt(ktS)*lambdaEff
    );

     volScalarField Pkt(nuts*S2);

     volScalarField ktL(kt_ - ktS);
     volScalarField ReOmega(sqr(y_)*Omega/nu());
     volScalarField nutl // Large scale eddy viscosity
    (
        min
        (
            C11_*fTaul(lambdaEff, ktL, Omega)*Omega*sqr(lambdaEff)
           *sqrt(ktL)*lambdaEff/nu()
          + C12_*BetaTS(ReOmega)*ReOmega*sqr(y_)*Omega
        ,
            0.5*(kl_ + ktL)/(sqrt(S2) + omegaMin_)
        )
    );

     volScalarField Pkl(nutl*S2);

     volScalarField alphaTEff
    (
        alphaT(lambdaEff, fv(sqr(fw)*kt_/nu()/(omega_ + omegaMin_)), ktS)
    );

    // By pass source term divided by kl_

     dimensionedScalar fwMin("SMALL", dimless, ROOTVSMALL);

     volScalarField Rbp
    (
        CR_*(1.0 - exp(-phiBP(Omega)()/Abp_))*omega_
       /(fw + fwMin)
    );

     volScalarField fNatCrit(1.0 - exp(-Cnc_*sqrt(kl_)*y_/nu()));
```

```
    // Natural source term divided by kl_
     volScalarField Rnat
    (
        CrNat_*(1.0 - exp(-phiNAT(ReOmega, fNatCrit)/Anat_))*Omega
    );

     volScalarField Dl(nu()*magSqr(fvc::grad(sqrt(kl_))));

     volScalarField Dt(nu()*magSqr(fvc::grad(sqrt(kt_))));

    //// DES length determining function

    volScalarField FDES(this->FDES());

////////////////////// Test and output variables /////////////////

    FDESout_ = FDES;



////////////////////// MODEL EQUATIONS ///////////////////////////

    // Turbulent kinetic energy equation
    {
        fvScalarMatrix ktEqn
        (
        fvm::ddt(kt_)
      + fvm::div(phi_, kt_)
      - fvm::laplacian(DkEff(alphaTEff), kt_, "laplacian(alphaTEff,kt)")
     ==
        Pkt
      + (Rbp + Rnat)*kl_
      - Dt //MAYBE NEEDED HERE? AS FDES*DT?
      - fvm::Sp(FDES*omega_, kt_)
        );

        ktEqn.relax();
        ktEqn.solve();
    }
    bound(kt_, kMin_);

    // Turbulent frequency equation
    {
        fvScalarMatrix omegaEqn
        (
fvm::ddt(omega_)
      + fvm::div(phi_, omega_)
      - fvm::laplacian(DomegaEff(alphaTEff), omega_)
     ==
Cw1_*Pkt*omega_/(kt_ + kMin_)
      - fvm::SuSp
(
    (1.0 - CwR_/(fw + fwMin))*kl_*(Rbp + Rnat)/(kt_ + kMin_)
  , omega_
)
      - fvm::Sp(Cw2_*sqr(fw)*omega_, omega_)
      + Cw3_*fOmega(lambdaEff, lambdaT)*alphaTEff*sqr(fw)*sqrt(kt_)/pow3(y_)
        );
```

```
        omegaEqn.relax();
        omegaEqn.solve();
    }
    bound(omega_, omegaMin_);

    // laminar kinetic energy equation
    {
        fvScalarMatrix klEqn
        (
fvm::ddt(kl_)
      + fvm::div(phi_, kl_)
      - fvm::laplacian(nu(), kl_)
     ==
Pkl
      - fvm::Sp(Rbp + Rnat + Dl/(kl_ + kMin_), kl_)
    );

        klEqn.relax();
        klEqn.solve();
    }
    bound(kl_, kMin_);

    // include new definition for nutsE and nutlE
    //updateSubGridScaleFields(nutsE, nutlE);

    updateSubGridScaleFields(nuts, nutl);
}


//// class functions ////////////////

tmp<volScalarField> kkLOmegaDES::epsilon() const
{
    return 2.0*nuEff()*magSqr(symm(fvc::grad(U())));
}


tmp<volSymmTensorField> kkLOmegaDES::B() const
{
    return ((2.0/3.0)*I)*k() - nuSgs()*twoSymm(fvc::grad(U()));
}


tmp<volSymmTensorField> kkLOmegaDES::devReff() const
{
    return -nuEff()*dev(twoSymm(fvc::grad(U())));
}


tmp<fvVectorMatrix> kkLOmegaDES::divDevReff(volVectorField& U) const
{
    return
    (
      - fvm::laplacian(nuEff(), U)
      - fvc::div(nuEff()*dev(T(fvc::grad(U))))
    );
}
```

```
tmp<fvVectorMatrix> kkLOmegaDES::divDevRhoReff
(
    const volScalarField& rho,
    volVectorField& U
) const
{
    volScalarField muEff("muEff", rho*nuEff());

    return
    (
      - fvm::laplacian(muEff, U)
      - fvc::div(muEff*dev(T(fvc::grad(U))))
    );
}


bool kkLOmegaDES::read()
{
    if (LESModel::read())
    {
        A0_.readIfPresent(coeffDict());
        As_.readIfPresent(coeffDict());
        Av_.readIfPresent(coeffDict());
        Abp_.readIfPresent(coeffDict());
        Anat_.readIfPresent(coeffDict());
        Abp_.readIfPresent(coeffDict());
        Ats_.readIfPresent(coeffDict());
        CbpCrit_.readIfPresent(coeffDict());
        Cnc_.readIfPresent(coeffDict());
        CnatCrit_.readIfPresent(coeffDict());
        Cint_.readIfPresent(coeffDict());
        CtsCrit_.readIfPresent(coeffDict());
        CrNat_.readIfPresent(coeffDict());
        C11_.readIfPresent(coeffDict());
        C12_.readIfPresent(coeffDict());
        CR_.readIfPresent(coeffDict());
        CalphaTheta_.readIfPresent(coeffDict());
        Css_.readIfPresent(coeffDict());
        CtauL_.readIfPresent(coeffDict());
        Cw1_.readIfPresent(coeffDict());
        Cw2_.readIfPresent(coeffDict());
        Cw3_.readIfPresent(coeffDict());
        CwR_.readIfPresent(coeffDict());
        Clambda_.readIfPresent(coeffDict());
        CmuStd_.readIfPresent(coeffDict());
        Prtheta_.readIfPresent(coeffDict());
        Sigmak_.readIfPresent(coeffDict());
        Sigmaw_.readIfPresent(coeffDict());
CDES_.readIfPresent(coeffDict());
kMin_.readIfPresent(*this);
omegaMin_.readIfPresent(*this);

        return true;
    }
    else
    {
        return false;
    }
```

```
}


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace LESModels
} // End namespace incompressible
} // End namespace Foam

// ************************************************************************* //
```

Appendix D

The $k_T - k_L - \omega$ DDES model OpenFOAM implementation

D.1 kklOmegaDDES.H

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2015 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Class
    Foam::incompressible::LESModels::kkLOmegaDDES

Group
    grpIcoLESTurbulence

Description
    k-kl-Omega-DDES turbulence model for incompressible flows.

        The baseline RAS turbulence model used is described in:

    \verbatim
        Walters, D. K., & Cokljat, D. (2008).
```

```
        A three-equation eddy-viscosity model for Reynolds-averaged
        Navier-Stokes simulations of transitional flow.
        Journal of Fluids Engineering, 130(12), 121401.
    \endverbatim

    However the paper contains several errors which must be corrected for the
    RAS model to operate correctly as explained in

       \verbatim
        Medina, H., & Early, J. (2014).
        Modelling transition due to backward-facing steps using the laminar
        kinetic energy concept.
        European Journal of Mechanics B/Fluids, 44(2014), 60-68.
    \endverbatim
-------------------------------------------------------------------------------
DISCLAIMER:
    This model is part of the requirements for the degree of Master by Research
    in Aerospace Egineering at Coventry University, under the supervision of
    Dr. Humberto Medina, Senior Lecturer in Aerospace Engineering.
-------------------------------------------------------------------------------

SourceFiles
    kkLOmegaDDES.C

\*---------------------------------------------------------------------------*/

#ifndef kkLOmegaDDES_H
#define kkLOmegaDDES_H

#include "LESModel.H"
#include "volFields.H"
#include "wallDist.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
namespace incompressible
{
namespace LESModels
{

/*---------------------------------------------------------------------------*\
                        Class kkLOmegaDDES Declaration
\*---------------------------------------------------------------------------*/

class kkLOmegaDDES
:
    public LESModel
{
    // Private Member Functions

        //- Update sub-grid scale fields
        void updateSubGridScaleFields(const volScalarField& nuts, const volScalarField& nutl);

        // Disallow default bitwise copy construct and assignment
        kkLOmegaDDES(const kkLOmegaDDES&);
        kkLOmegaDDES& operator=(const kkLOmegaDDES&);
```

```
protected:

    // Protected Member Functions

// kklOmega Private member functions

        tmp<volScalarField> fv(const volScalarField& Ret) const;

        tmp<volScalarField> fINT() const;

        tmp<volScalarField> fSS(const volScalarField& omega) const;

        tmp<volScalarField> Cmu(const volScalarField& S) const;

        tmp<volScalarField> BetaTS(const volScalarField& ReOmega) const;

        tmp<volScalarField> fTaul
        (
            const volScalarField& lambdaEff,
            const volScalarField& ktL,
            const volScalarField& omega
        ) const;

        tmp<volScalarField> alphaT
        (
            const volScalarField& lambdaEff,
            const volScalarField& fv,
            const volScalarField& ktS
        ) const;

        tmp<volScalarField> fOmega
        (
            const volScalarField& lambdaEff,
            const volScalarField& lambdaT
        ) const;

        tmp<volScalarField> phiBP(const volScalarField& omega) const;

        tmp<volScalarField> phiNAT
        (
            const volScalarField& ReOmega,
            const volScalarField& fNatCrit
        ) const;

tmp<volScalarField> FDDES(const volScalarField& FS) const;

         tmp<volScalarField> F1(const volScalarField& CDkOmega) const;

        // Model constants
dimensionedScalar alphaOmega2_;
            dimensionedScalar A0_;
            dimensionedScalar As_;
            dimensionedScalar Av_;
            dimensionedScalar Abp_;
            dimensionedScalar Anat_;
            dimensionedScalar Ats_;
            dimensionedScalar CbpCrit_;
            dimensionedScalar Cnc_;
```

```
            dimensionedScalar CnatCrit_;
            dimensionedScalar Cint_;
            dimensionedScalar CtsCrit_;
            dimensionedScalar CrNat_;
            dimensionedScalar C11_;
            dimensionedScalar C12_;
            dimensionedScalar CR_;
            dimensionedScalar CalphaTheta_;
            dimensionedScalar Css_;
            dimensionedScalar CtauL_;
            dimensionedScalar Cw1_;
            dimensionedScalar Cw2_;
            dimensionedScalar Cw3_;
            dimensionedScalar CwR_;
            dimensionedScalar Clambda_;
            dimensionedScalar CmuStd_;
            dimensionedScalar Prtheta_;
            dimensionedScalar Sigmak_;
            dimensionedScalar Sigmaw_;
    dimensionedScalar CDES_;
    dimensionedScalar kMin_;
    dimensionedScalar omegaMin_;

        // Fields

            volScalarField kt_;
            volScalarField kl_;
            volScalarField omega_;
            volScalarField nuSgs_;
            volScalarField FDDESout_;


            //- Wall distance
            wallDist y_;


public:

    //- Runtime type information
    TypeName("kkLOmegaDDES");


    // Constructors

        //- Construct from components
        kkLOmegaDDES
        (
            const volVectorField& U,
            const surfaceScalarField& phi,
            transportModel& transport,
            const word& turbulenceModelName = turbulenceModel::typeName,
            const word& modelName = typeName
        );


    //- Destructor
    virtual ~kkLOmegaDDES()
    {}
```

```
// Member Functions

    //- Return the effective diffusivity for k
    tmp<volScalarField> DkEff(const volScalarField& alphaT) const
    {
        return tmp<volScalarField>
        (
            new volScalarField("DkEff", alphaT/Sigmak_ + nu())
        );
    }

    //- Return the effective diffusivity for omega
    tmp<volScalarField> DomegaEff(const volScalarField& alphaT) const
    {
        return tmp<volScalarField>
        (
            new volScalarField("DomegaEff", alphaT/Sigmaw_ + nu())
        );
    }

    //- Return SGS kinetic energy
    virtual tmp<volScalarField> k() const
    {
        return (kt_ + kl_); // this needs changing!!!!!
    }

    //- Return the laminar kinetic energy
    virtual tmp<volScalarField> kl() const
    {
        return kl_;
    }

    //- Return the turbulence kinetic energy
    virtual tmp<volScalarField> kt() const
    {
        return kt_;
    }

    //- Return the turbulence specific dissipation rate
    virtual tmp<volScalarField> omega() const
    {
        return omega_;
    }


    //- Return sub-grid disipation rate
    virtual tmp<volScalarField> epsilon() const;

    //- Return SGS viscosity
    virtual tmp<volScalarField> nuSgs() const
    {
        return nuSgs_;
    }

    //- Return the sub-grid stress tensor.
    virtual tmp<volSymmTensorField> B() const;

    //- Return the effective sub-grid turbulence stress tensor
```

```
        //  including the laminar stress
        virtual tmp<volSymmTensorField> devReff() const;

        //- Return the deviatoric part of the effective sub-grid
        //  turbulence stress tensor including the laminar stress
        virtual tmp<fvVectorMatrix> divDevReff(volVectorField& U) const;

        //- Return the deviatoric part of the effective sub-grid
        //  turbulence stress tensor including the laminar stress
        virtual tmp<fvVectorMatrix> divDevRhoReff
        (
            const volScalarField& rho,
            volVectorField& U
        ) const;

        //- Solve the turbulence equations (k-w) and correct the turbulence
        //  viscosity
        virtual void correct(const tmp<volTensorField>& gradU);

        //- Read LESProperties dictionary
        virtual bool read();
};


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace LESModels
} // End namespace incompressible
} // End namespace Foam

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //
```

D.2 kklOmegaDDES.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2013 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------------*/

#include "kkLOmegaDDES.H"
#include "addToRunTimeSelectionTable.H"
#include "wallDist.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
namespace incompressible
{
namespace LESModels
{

// * * * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * * * //

defineTypeNameAndDebug(kkLOmegaDDES, 0);
addToRunTimeSelectionTable(LESModel, kkLOmegaDDES, dictionary);

// * * * * * * * * * * * * * * * Protected Member Functions  * * * * * * * * * * * * //

void kkLOmegaDDES::updateSubGridScaleFields(const volScalarField& nuts, const volScalarField& nutl)
{
    nuSgs_ = nuts + nutl;
    nuSgs_.correctBoundaryConditions();
}


/// kklOmega Fuctions

tmp<volScalarField> kkLOmegaDDES::fv(const volScalarField& Ret) const
{
    return(1.0 - exp(-sqrt(Ret)/Av_));
```

```
}


tmp<volScalarField> kkLOmegaDDES::fINT() const
{
    return
    (
        min
        (
            kt_/(Cint_*(kl_ + kt_ + kMin_)),
            dimensionedScalar("1.0", dimless, 1.0)
        )
    );
}


tmp<volScalarField> kkLOmegaDDES::fSS(const volScalarField& Omega) const
{
    return(exp(-sqr(Css_*nu()*Omega/(kt_ + kMin_))));
}


tmp<volScalarField> kkLOmegaDDES::Cmu(const volScalarField& S) const
{
    return(1.0/(A0_ + As_*(S/(omega_ + omegaMin_))));
}


tmp<volScalarField> kkLOmegaDDES::BetaTS(const volScalarField& ReOmega) const
{
    return(scalar(1) - exp(-sqr(max(ReOmega - CtsCrit_, scalar(0)))/Ats_));
}


tmp<volScalarField> kkLOmegaDDES::fTaul
(
    const volScalarField& lambdaEff,
    const volScalarField& ktL,
    const volScalarField& Omega
) const
{
    return
    (
        scalar(1)
      - exp
        (
            -CtauL_*ktL
          /
            (
                sqr
                (
                    lambdaEff*Omega
                  + dimensionedScalar
                    (
                        "ROOTVSMALL",
                        dimLength*inv(dimTime),
                        ROOTVSMALL
                    )
                )
```

```
            )
        )
    );
}


tmp<volScalarField> kkLOmegaDDES::alphaT
(
    const volScalarField& lambdaEff,
    const volScalarField& fv,
    const volScalarField& ktS
) const
{
    return(fv*CmuStd_*sqrt(ktS)*lambdaEff);
}


tmp<volScalarField> kkLOmegaDDES::fOmega
(
    const volScalarField& lambdaEff,
    const volScalarField& lambdaT
) const
{
    return
    (
        scalar(1)
      - exp
        (
           -0.41
           *pow4
            (
                lambdaEff
              / (
                    lambdaT
                  + dimensionedScalar
                    (
                        "ROTVSMALL",
                        lambdaT.dimensions(),
                        ROOTVSMALL
                    )
                )
            )
        )
    );
}


tmp<volScalarField> kkLOmegaDDES::phiBP(const volScalarField& Omega) const
{
    return
    (
        min
        (
            max
            (
                kt_/nu()
             / (
                    Omega
                  + dimensionedScalar
```

```
                    (
                        "ROTVSMALL",
                        Omega.dimensions(),
                        ROOTVSMALL
                    )
                )
              - CbpCrit_,
                scalar(0)
            ),
            scalar(50.0)
        )
    );
}


tmp<volScalarField> kkLOmegaDDES::phiNAT
(
    const volScalarField& ReOmega,
    const volScalarField& fNatCrit
) const
{
    return
    (
        max
        (
            ReOmega
          - CnatCrit_
          / (
                fNatCrit + dimensionedScalar("ROTVSMALL", dimless, ROOTVSMALL)
            ),
            scalar(0)
        )
    );
}

// F_DDES term definition

tmp<volScalarField> kkLOmegaDDES::FDDES(const volScalarField& FS) const
{
    return max
    (
     sqrt(kt_)/(CDES_*omega_*delta())*(scalar(1) - FS),
scalar(1)
    );
}

tmp<volScalarField> kkLOmegaDDES::F1(const volScalarField& CDkOmega) const
{
    tmp<volScalarField> CDkOmegaPlus = max
    (
        CDkOmega,
        dimensionedScalar("1.0e-10", dimless/sqr(dimTime), 1.0e-10)
    );

    tmp<volScalarField> arg1 = min
    (
        min
        (
            max
```

```
            (
                //(sqrt(kt_)/(omega_*y_)), // ORIGINAL LINE
                //CmuStd_*scalar(500)*nu()/(sqr(y_)*omega_) // ORIGINAL LINE
                (sqrt(kt_)/(CmuStd_*omega_*y_)),
                scalar(500)*nu()/(sqr(y_)*omega_)
            ),
    //(4*alphaOmega2_)*kt_/(CDkOmegaPlus*sqr(y_)* CmuStd_) //ORIGINAL LINE
        (4*alphaOmega2_)*kt_/(CDkOmegaPlus*sqr(y_))
        ),
        scalar(10)
    );

    return tanh(pow4(arg1));
}


// * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * * //

kkLOmegaDDES::kkLOmegaDDES
(
    const volVectorField& U,
    const surfaceScalarField& phi,
    transportModel& transport,
    const word& turbulenceModelName,
    const word& modelName
)
:
    LESModel(modelName, U, phi, transport, turbulenceModelName),

    alphaOmega2_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "alphaOmega2",
            coeffDict_,
     //0.5 //This coefficient worked
     0.8547
        )
    ),
    A0_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "A0",
            coeffDict_,
            4.04
        )
    ),
    As_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "As",
            coeffDict_,
            2.12
        )
    ),
    Av_
    (
```

```
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Av",
            coeffDict_,
            6.75
        )
    ),
    Abp_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Abp",
            coeffDict_,
            0.6
        )
    ),
    Anat_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Anat",
            coeffDict_,
            200
        )
    ),
    Ats_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Ats",
            coeffDict_,
            200
        )
    ),
    CbpCrit_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CbpCrit",
            coeffDict_,
            1.2
        )
    ),
    Cnc_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cnc",
            coeffDict_,
            0.1
        )
    ),
    CnatCrit_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CnatCrit",
            coeffDict_,
            1250
```

```
        )
    ),
    Cint_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cint",
            coeffDict_,
            0.75
        )
    ),
    CtsCrit_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CtsCrit",
            coeffDict_,
            1000
        )
    ),
    CrNat_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CrNat",
            coeffDict_,
            0.02
        )
    ),
    C11_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "C11",
            coeffDict_,
            3.4e-6
        )
    ),
    C12_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "C12",
            coeffDict_,
            1.0e-10
        )
    ),
    CR_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CR",
            coeffDict_,
            0.12
        )
    ),
    CalphaTheta_
    (
        dimensioned<scalar>::lookupOrAddToDict
```

```
        (
            "CalphaTheta",
            coeffDict_,
            0.035
        )
    ),
    Css_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Css",
            coeffDict_,
            1.5
        )
    ),
    CtauL_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CtauL",
            coeffDict_,
            4360
        )
    ),
    Cw1_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cw1",
            coeffDict_,
            0.44
        )
    ),
    Cw2_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cw2",
            coeffDict_,
            0.92
        )
    ),
    Cw3_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cw3",
            coeffDict_,
            0.3
        )
    ),
    CwR_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CwR",
            coeffDict_,
            1.5
        )
```

```
    ),
    Clambda_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Clambda",
            coeffDict_,
            2.495
        )
    ),
    CmuStd_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CmuStd",
            coeffDict_,
            0.09
        )
    ),
    Prtheta_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Prtheta",
            coeffDict_,
            0.85
        )
    ),
    Sigmak_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Sigmak",
            coeffDict_,
            1
        )
    ),
    Sigmaw_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Sigmaw",
            coeffDict_,
            1.17
        )
    ),

    CDES_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CDES",
            coeffDict_,
            0.61
        )
    ),

    kMin_("kMin", sqr(dimVelocity), SMALL),
```

```
    omegaMin_("omegaMin", dimless/dimTime, SMALL),

    kt_
    (
        IOobject
        (
            "kt",
            runTime_.timeName(),
            mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_
    ),

    kl_
    (
        IOobject
        (
            "kl",
            runTime_.timeName(),
            mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_
    ),

    omega_
    (
        IOobject
        (
            "omega",
            runTime_.timeName(),
            mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_
    ),

    nuSgs_
    (
        IOobject
        (
            "nuSgs",
            runTime_.timeName(),
            mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_
    ),

    FDDESout_
    (
        IOobject
        (
            "FDDES",
```

```
            runTime_.timeName(),
            mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_,
        dimensionedScalar("FDDES", dimless, 0.0)
    ),

    y_(mesh_)


{
    kMin_.readIfPresent(*this);
    omegaMin_.readIfPresent(*this);

    bound(kt_, kMin_);
    bound(kl_, kMin_);
    bound(omega_, omegaMin_);

    updateSubGridScaleFields(kt_/max(omega_, omegaMin_), kl_/max(omega_, omegaMin_));

    printCoeffs();
}


// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * * //

void kkLOmegaDDES::correct(const tmp<volTensorField>& gradU)
{
    LESModel::correct(gradU);

    if (mesh_.changing())
    {
        y_.correct();
    }

    volScalarField S2(2.0*magSqr(symm(gradU())));
    volScalarField Omega(sqrt(2.0)*mag(skew(gradU())));
    gradU.clear();



/// kklOmega functions and definitions

     volScalarField lambdaT(sqrt(kt_)/(omega_ + omegaMin_));

     volScalarField lambdaEff(min(Clambda_*y_, lambdaT));

     volScalarField fw
    (
        pow
        (
            lambdaEff
           /(lambdaT + dimensionedScalar("SMALL", dimLength, ROOTVSMALL)),
            2.0/3.0
        )
    );
```

```
 volScalarField ktS(fSS(Omega)*fw*kt_);

 volScalarField nuts // Small scale eddy viscosity
(
    fv(sqr(fw)*kt_/nu()/(omega_ + omegaMin_))
   *fINT()
   *Cmu(sqrt(S2))*sqrt(ktS)*lambdaEff
);

 volScalarField Pkt(nuts*S2);

 volScalarField ktL(kt_ - ktS);
 volScalarField ReOmega(sqr(y_)*Omega/nu());
 volScalarField nutl // Large scale eddy viscosity
(
    min
    (
        C11_*fTaul(lambdaEff, ktL, Omega)*Omega*sqr(lambdaEff)
       *sqrt(ktL)*lambdaEff/nu()
      + C12_*BetaTS(ReOmega)*ReOmega*sqr(y_)*Omega
    ,
        0.5*(kl_ + ktL)/(sqrt(S2) + omegaMin_)
    )
);

 volScalarField Pkl(nutl*S2);

 volScalarField alphaTEff
(
    alphaT(lambdaEff, fv(sqr(fw)*kt_/nu()/(omega_ + omegaMin_)), ktS)
);

// By pass source term divided by kl_

 dimensionedScalar fwMin("SMALL", dimless, ROOTVSMALL);

 volScalarField Rbp
(
    CR_*(1.0 - exp(-phiBP(Omega)()/Abp_))*omega_
   /(fw + fwMin)
);

 volScalarField fNatCrit(1.0 - exp(-Cnc_*sqrt(kl_)*y_/nu()));

// Natural source term divided by kl_
 volScalarField Rnat
(
    CrNat_*(1.0 - exp(-phiNAT(ReOmega, fNatCrit)/Anat_))*Omega
);

 volScalarField Dl(nu()*magSqr(fvc::grad(sqrt(kl_))));

 volScalarField Dt(nu()*magSqr(fvc::grad(sqrt(kt_))));

 // Functions SST (Menter's F1 blending function)

volScalarField CDkOmega
(
    (2*alphaOmega2_)*(fvc::grad(kt_) & fvc::grad(omega_))/max(omega_ , omegaMin_)
```

```
    );

    volScalarField F1(this->F1(CDkOmega));


//// DES length determining function

    volScalarField FDDES(this->FDDES(F1));

////////////////////// Test and output variables /////////////////

    FDDESout_ = FDDES;



////////////////////// MODEL EQUATIONS ////////////////////////////

    // Turbulent kinetic energy equation
    {
        fvScalarMatrix ktEqn
        (
        fvm::ddt(kt_)
      + fvm::div(phi_, kt_)
      - fvm::laplacian(DkEff(alphaTEff), kt_, "laplacian(alphaTEff,kt)")
     ==
        Pkt
      + (Rbp + Rnat)*kl_
      - Dt //MAYBE NEEDED HERE? AS FDDES*DT?
      - fvm::Sp(FDDES*omega_, kt_)
        );

        ktEqn.relax();
        ktEqn.solve();
    }
    bound(kt_, kMin_);

    // Turbulent frequency equation
    {
        fvScalarMatrix omegaEqn
        (
fvm::ddt(omega_)
      + fvm::div(phi_, omega_)
      - fvm::laplacian(DomegaEff(alphaTEff), omega_)
     ==
Cw1_*Pkt*omega_/(kt_ + kMin_)
      - fvm::SuSp
(
    (1.0 - CwR_/(fw + fwMin))*kl_*(Rbp + Rnat)/(kt_ + kMin_)
  , omega_
)
      - fvm::Sp(Cw2_*sqr(fw)*omega_, omega_)
      + Cw3_*fOmega(lambdaEff, lambdaT)*alphaTEff*sqr(fw)*sqrt(kt_)/pow3(y_)
        );

        omegaEqn.relax();
        omegaEqn.solve();
    }
    bound(omega_, omegaMin_);
```

```
    // laminar kinetic energy equation
    {
        fvScalarMatrix klEqn
        (
fvm::ddt(kl_)
      + fvm::div(phi_, kl_)
      - fvm::laplacian(nu(), kl_)
     ==
Pkl
      - fvm::Sp(Rbp + Rnat + Dl/(kl_ + kMin_), kl_)
    );

        klEqn.relax();
        klEqn.solve();
    }
    bound(kl_, kMin_);

    // include new definition for nutsE and nutlE
    //updateSubGridScaleFields(nutsE, nutlE);

    updateSubGridScaleFields(nuts, nutl);
}


//// class functions ///////////////

tmp<volScalarField> kkLOmegaDDES::epsilon() const
{
    return 2.0*nuEff()*magSqr(symm(fvc::grad(U())));
}


tmp<volSymmTensorField> kkLOmegaDDES::B() const
{
    return ((2.0/3.0)*I)*k() - nuSgs()*twoSymm(fvc::grad(U()));
}


tmp<volSymmTensorField> kkLOmegaDDES::devReff() const
{
    return -nuEff()*dev(twoSymm(fvc::grad(U())));
}


tmp<fvVectorMatrix> kkLOmegaDDES::divDevReff(volVectorField& U) const
{
    return
    (
      - fvm::laplacian(nuEff(), U)
      - fvc::div(nuEff()*dev(T(fvc::grad(U))))
    );
}


tmp<fvVectorMatrix> kkLOmegaDDES::divDevRhoReff
(
    const volScalarField& rho,
    volVectorField& U
) const
```

```
{
    volScalarField muEff("muEff", rho*nuEff());

    return
    (
      - fvm::laplacian(muEff, U)
      - fvc::div(muEff*dev(T(fvc::grad(U))))
    );
}


bool kkLOmegaDDES::read()
{
    if (LESModel::read())
    {
     alphaOmega2_.readIfPresent(coeffDict());
        A0_.readIfPresent(coeffDict());
        As_.readIfPresent(coeffDict());
        Av_.readIfPresent(coeffDict());
        Abp_.readIfPresent(coeffDict());
        Anat_.readIfPresent(coeffDict());
        Abp_.readIfPresent(coeffDict());
        Ats_.readIfPresent(coeffDict());
        CbpCrit_.readIfPresent(coeffDict());
        Cnc_.readIfPresent(coeffDict());
        CnatCrit_.readIfPresent(coeffDict());
        Cint_.readIfPresent(coeffDict());
        CtsCrit_.readIfPresent(coeffDict());
        CrNat_.readIfPresent(coeffDict());
        C11_.readIfPresent(coeffDict());
        C12_.readIfPresent(coeffDict());
        CR_.readIfPresent(coeffDict());
        CalphaTheta_.readIfPresent(coeffDict());
        Css_.readIfPresent(coeffDict());
        CtauL_.readIfPresent(coeffDict());
        Cw1_.readIfPresent(coeffDict());
        Cw2_.readIfPresent(coeffDict());
        Cw3_.readIfPresent(coeffDict());
        CwR_.readIfPresent(coeffDict());
        Clambda_.readIfPresent(coeffDict());
        CmuStd_.readIfPresent(coeffDict());
        Prtheta_.readIfPresent(coeffDict());
        Sigmak_.readIfPresent(coeffDict());
        Sigmaw_.readIfPresent(coeffDict());
CDES_.readIfPresent(coeffDict());
kMin_.readIfPresent(*this);
omegaMin_.readIfPresent(*this);

        return true;
    }
    else
    {
        return false;
    }
}


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
} // End namespace LESModels
} // End namespace incompressible
} // End namespace Foam

// ************************************************************************* //
```